ESD-TR-77-127

MTR-3365

SPECIAL - PURPOSE PROCESSORS FOR SURVEILLANCE RADAR APPLICATIONS

MAY 1977

Prepared for

DEPUTY FOR DEVELOPMENT PLANS
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
Hanscom Air Force Base, Bedford, Massachusetts

Project No. 7010
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract No. F19628-76-C-0001

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

William W. Selah

WILLIAM W. SELAH, 2dLt, USAF
Project Officer
Technological Planning
Deputy for Development Plans

John B. Tindall

JOHN B. TINDALL, Lt Colonel, USAF
Director, Technological Planning
Deputy for Development Plans

Hugh M. Miller

HUGH M. MILLER, Colonel, USAF
Assistant Deputy for Development Plans

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ESD-TR-77-127	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): SPECIAL—PURPOSE PROCESSORS FOR SURVEILLANCE RADAR APPLICATIONS	5. TYPE OF REPORT & PERIOD COVERED
6. PERFORMING ORG. REPORT NUMBER: MTR-3365	
7. AUTHOR(s): R.W. Jacobus, V. Kross	8. CONTRACT OR GRANT NUMBER(s): F19628-76-C-0001
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The MITRE Corporation, P.O. Box 208, Bedford, MA 01730	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Project No. 7010
11. CONTROLLING OFFICE NAME AND ADDRESS: Deputy for Development Plans, Electronic Systems Division, AFSC, Hanscom Air Force Base, Bedford, MA 01731	12. REPORT DATE: MAY 1977
	13. NUMBER OF PAGES: 82
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

COORDINATE TRANSFORMATION
LOW COST
MICROPROCESSOR
SATELLITE PROCESSOR
SPECIAL-PURPOSE PROCESSOR
TRACKING

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Many track-while-scan surveillance systems utilize a large central computer to correlate the incoming sensor data with established tracks on targets of interest. These correlations must be done in near-real time at a high data rate and often consume a substantial fraction of the computer's CPU capacity; as a consequence, both the computer hardware and its software are expensive. Using the Joint

(over)

20. Abstract (concluded)

Surveillance System (JSS), a radar network for continental air defense as a specific example, a special-purpose processor is described. It can be connected as a "satellite" to virtually any computer and can relieve the computer of the coordinate-conversion and track-correlation burden. The impact of the processor on reducing the hardware and software costs of the overall system is discussed.

ACKNOWLEDGMENT

This report has been prepared by The MITRE Corporation under Project No. 7010. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION I

INTRODUCTION

The high cost of military electronics has focussed attention at all levels on the possibility of using technology to lower the life-cycle costs of systems. The problems are enormously complicated by the multitude of interactions and interfaces that must be accommodated, by built-in costs that cannot be altered appreciably, and by the lack of confidence in new approaches to the design and procurement of equipment. A particularly severe problem is found in computer-related subsystems; the present costs of both computer equipment and computer software are much too high. Software is assuming an increasingly larger portion of the computer system costs, despite such innovations as "structured programming" and other similar means to discipline the design and debugging process. The recent availability of low-cost mini-computers and micro-computers has raised the hope of decentralized computational facilities, and a corresponding lowering of the overall cost of the computer system components of large military systems.

Another very promising approach to lowering the cost of computation is the special-purpose "satellite" processor. It is not a solution to all computer problems, but it does apply to the requirements of many systems, and should be considered seriously during the conceptual design phase.

The satellite processor is defined here as a device that can be attached to a general-purpose digital computer, using one or more of

the standard input/output peripheral channels of the computer. The satellite is designed to perform some well-defined portion of the system's computational load. It can utilize hard-wired digital equipment for the execution of fixed algorithms, and it can contain general-purpose computer elements (e.g., micro-processors) as well. Ordinarily it is not in series with the data stream, as might be the case with some hard-wired processors, but rather sits off to the side and manipulates data that have already been handled by the computer.

The satellite's standard function is to process tables of data stored in the computer's memory. The computer programmer arranges to have the data available in a certain set of consecutive memory locations; he "calls" the satellite, and at some later time he can find the processed table occupying either the original locations or some other pre-arranged addresses. With careful design of the satellite, it can be made very simple for the programmer to use, and can present a minimum of interface problems for the computer hardware designer. From the hardware point of view, the processor should appear to the computer as a standard peripheral device, such as an analog-to-digital converter or a magnetic disk memory; usually it is connected to a direct-memory-access (DMA) channel to permit the highest data-transfer rates. From the software programmer's point of view, the satellite should be "set up" with a minimum number of commands, should require no programmer attention or monitoring while it is performing its tasks, and should present a minimum load on the

computer's normal functions; e.g., the satellite should not require a maximum-rate data transfer of large blocks of data that would effectively paralyze the computer's central processor for some period of time.

Satellite processors are not a new concept. Examples of commercially available devices in this class are the Fast Fourier Transform processor or other similar array processors, and units which rapidly compute transcendental functions.

The satellite processor can represent a powerful tool in reducing both the cost and the risk of digital computation in a system. In many cases the use of one or more satellites can greatly change the architecture of the overall system, and often can result in large direct savings in conventional computer hardware.

Substantial software savings can often be obtained also, but they are somewhat more indirect and are heavily dependent upon both the hardware and software architecture utilized in the "conventional" approaches to the system design. Without satellite processors, the programmer is usually straining to make optimally efficient use of the computer's capability--by careful management of timing, use of complicated interrupt arrangements, overlaying of instructions and data, trading of memory for computation time, and searching for the shortest algorithms consistent with the desired accuracy. All of these activities make the programmer's task much more difficult, increase his tendency to make errors, and confuse the process of checking and debugging programs. By using satellite

processors, much of the computer's capacity is relieved, leaving the programmer with a relatively luxurious margin, and thereby permitting him to use "brute force" programming approaches that are easier to understand and document, and easier to test and debug.

These satellites can be quite inexpensive with respect to design costs, hardware implementation costs, and the indirect costs of programmer education. Since the satellite functions are generally well-defined and repetitive, the devices are easy to test as isolated units (i.e., not connected to the computer), and can be tested in a straightforward manner by the programmer in the operational real-time computer environment.

The comments made above about the relative virtues of satellite processors are, to a large extent, opinions held by the authors and their associates. To place these opinions in context, and to give them some additional credibility, we have selected an example of a satellite processor that performs track correlations and other tasks in a track-while-scan radar system. To be specific, so that detailed considerations can be examined, we have selected the Joint Surveillance System (JSS) as the particular radar application for treatment here.

SECTION II

THE JOINT SURVEILLANCE SYSTEM (JSS)

The Joint Surveillance System (JSS) is a network of surveillance radars located in the continental United States, Alaska, and Canada. JSS is an air-sovereignty system intended to replace the existing SAGE system, and will provide real-time information on all aircraft within this large surveillance region. Data from groups of about 25 radar sites are brought together at Regional Operations Control Centers (ROCC's), each containing large-scale data-processing facilities and provisions for approximately 30 operators' consoles.

Figure 2-1 shows a typical distribution of radars within control of a single ROCC; the circles have a radius of about 200 miles, which is representative of the surveillance range of the radars. Some of the sites are capable of measuring target range and azimuth only, while others have height-finders, and a few have modern "3-D" radars. In most cases, height information on a target is available only when requested; i.e., the ROCC must ask the radar to perform a height measurement on a particular target. All sites can interrogate the IFF transponders carried by friendly military and civilian aircraft. Data from the radars are transmitted to the ROCC's in a wide variety of standard formats. In all cases the basic radar information on aircraft targets is in polar form (range, azimuth, and height) expressed in the local coordinates of the surveillance site.

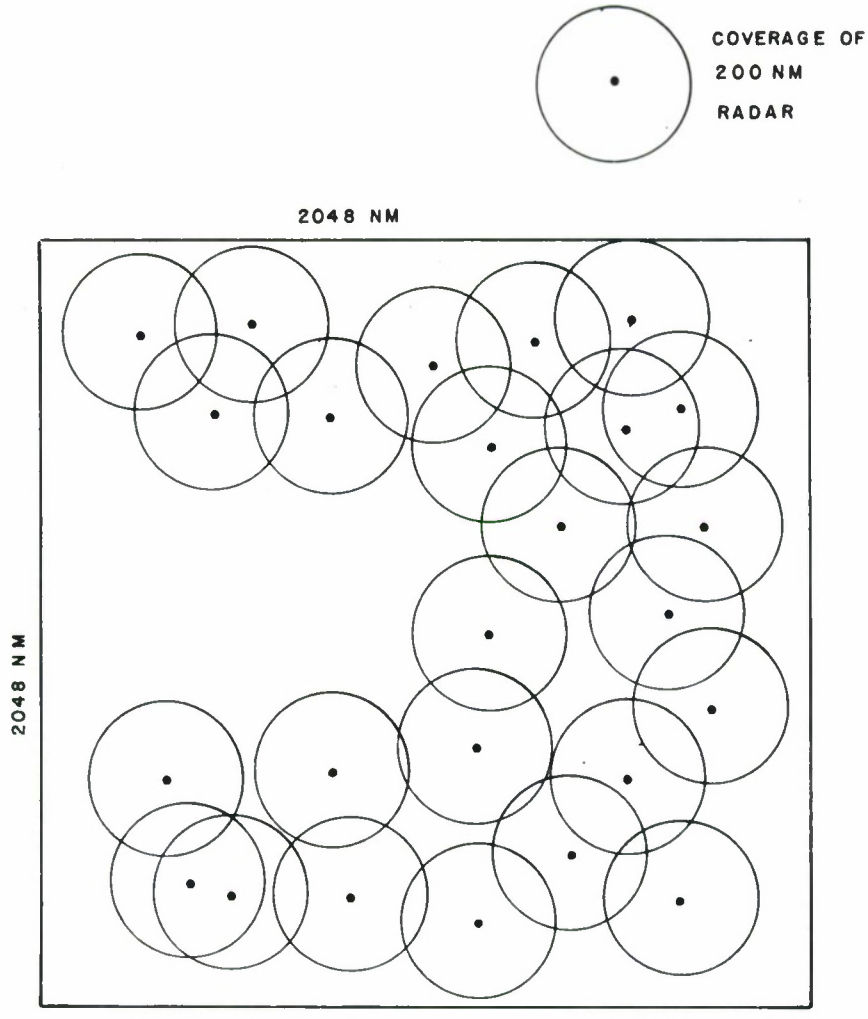


Figure 2-1 EXAMPLE OF JSS COVERAGE REGION

The data processor at a ROCC must perform many functions, among which are the following: It must maintain two-way communications with each of the 25 radar sites, and register all of the incoming target measurements with respect to a common reference grid, after converting from polar to rectangular coordinates. The processor must display all of the registered data at operator consoles, allow the operators to initiate tentative tracks, and then automatically track up to 200 targets. When necessary, computations for guiding an interceptor aircraft to an intersection with a given track must be performed. Track information and other data must be sent to and received from other ROCC's, the NORAD center, the Air Weather Service, and the FAA.

There are several reasonable options for the configuration of computing equipment at a ROCC. The simplest from a conceptual standpoint is a single large computer with sufficient speed and memory to perform all of the required tasks. Another option is to sub-divide the computational problem into several relatively independent tasks, each performed by a separate computer, as shown in Figure 2-2. On the basis of some informal studies conducted at MITRE, the multiple-computer option (where each computer is in the "medium-scale" class) represents a reasonable system architecture from the standpoint of hardware and software costs; the various separate tasks are each within the capability of medium-scale computers, the computational loads appear to be fairly well balanced, and the interfaces among the computers are relatively simple and straightforward. We cannot claim here that any multiple-

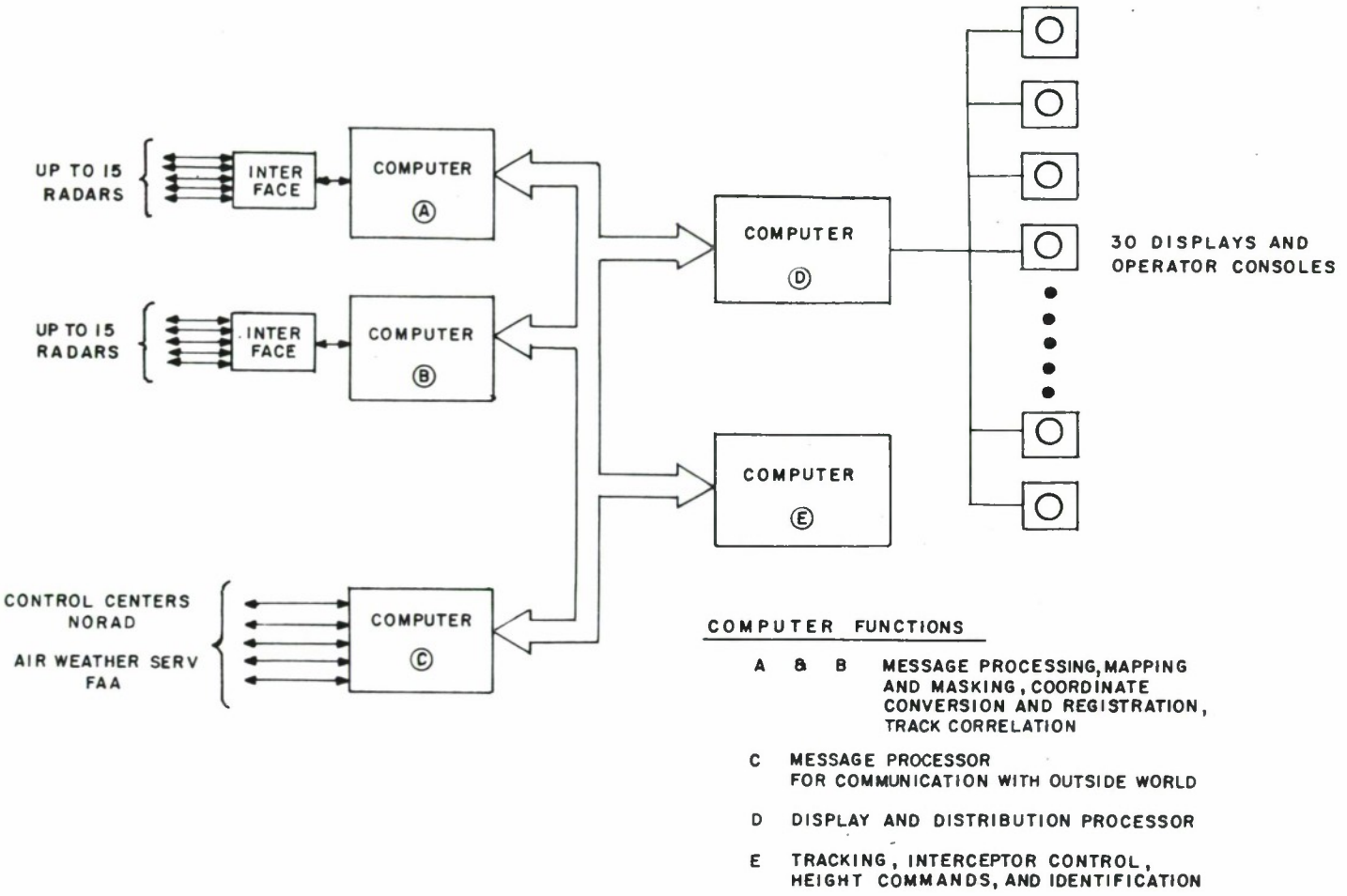


Figure 2-2 ONE CONFIGURATION FOR OVERALL JSS DATA PROCESSING SYSTEM

computer configuration is an optimum one for the JSS problem, and there is no reason to believe that the particular arrangement shown in Figure 2-2 will actually be implemented for JSS.

The arrangement of Figure 2-2 results in a physical separation of tasks, which has the advantage of simplifying our discussions in Section V about the cost impact of a special-purpose satellite. Therefore, for the remainder of this report we will use Figure 2-2 to represent the "before" configuration of a hypothetical version of JSS.

As shown in the figure, five computers are tied together with a bi-directional data bus (actually several independent busses). Two of the computers, labelled A and B, operate more or less in parallel to handle the high volume of raw data generated by the radars. Using special-purpose multiplex interface units, they bring the asynchronous radar data into temporary core storage. To limit the amount of data for subsequent processing, they perform various mapping and masking functions according to pre-set and/or operator-controlled thesholds and priority decisions. Although as many as 8500 radar reports may enter the ROCC in any six-second data frame, the result of the masking and mapping is to place a firm upper bound of 3600 reports per frame on the data passed on to the rest of the processing system. The two input computers perform polar-to-rectangular coordinate conversions on the data, and then provide "coarse" track correlation, as described below.

The large geographical extent of the surveillance volume for a ROCC places rather severe requirements on the accuracy of the coordinate conversion, and considerable effort has been expended in the JSS project to find a set of equations of minimum complexity which meet the requirements. If the raw data for a given radar report are represented as R (slant range), θ (azimuth), and H (height) then the following pair of equations may be used to compute the X- and Y-coordinates of the report:

$$X = K_4 + K_5 F \sin\alpha + K_6 F^2 \sin\beta$$

$$Y = K_7 + K_8 F \cos\alpha + K_9 F^2 \cos\beta$$

where $F = \sqrt{R^2 - (H-K_1)^2}$

$\alpha = \theta - K_2$

$\beta = 2\alpha - K_3$

The nine constants K_1 through K_9 are functions of the particular location of the radar site from which the report was obtained. Even these relatively simple equations can place a large burden on the computers, because two sines, two cosines, a square root, and at least seven multiplications must be performed for each of the 3600 reports in a data frame; special-purpose satellites which can rapidly compute sines and square roots may be essential adjuncts to the computers.

The process of "tracking" requires that one of the computers (labelled E in Figure 2-2) continuously updates a set of simple smoothing equations for each target. The track parameters are updated by comparing the predicted position of each target with the most recent radar reports. When a particular radar report lies close to the expected position of the target, the tracking algorithm assumes that the report is "correlated" with the target track, and updates the track parameters accordingly. Track correlation is thus a process of comparing the radar reports with the track predictions, and making decisions on the basis of the relative distances between them.

As mentioned earlier, the JSS must have the computational capacity to track up to 200 targets. If the computer were to use the simplest approach to correlation, it would have to test each of the 200 tracks against every one of the 3600 radar reports during any six-second data frame, or the equivalent of one correlation test every eight microseconds. Since this would clearly consume most of the CPU capacity of a typical computer, it is necessary for the JSS programmer to employ a variety of tricks or stratagems to avoid the brute-force testing of 3600 reports against 200 tracks. A part of the time-saving strategy is to first perform "coarse correlation" on the data, based on the notion that most of the 3600 reports are actually false alarms not correlated with any tracks; if the computer can recognize or tag those relatively few reports which are roughly correlated with the tracks, then the remaining reports can be ignored by the tracker.

Thus, using several types of sorting strategies which are not of direct interest here, the JSS programmer manages to make the following coarse test on each of the 3600 radar reports: Does the report lie within (say) 12 miles of the expected position of any track? We shall call this particular test the coarse track-correlation. Only those reports yielding an affirmative answer are passed on to the precision tracker which makes the final fine-grain correlations. In the ROCC configuration shown in Figure 2-2, input computers A and B perform the function of coarse track-correlation, while computer E performs final correlation and other related operations.

To complete this brief description and definition of the JSS computational requirements (emphasizing the data manipulations near the input portion of the ROCC), we must be more specific about the timing constraints in the system. The computations are organized into data frames which are six seconds in length, as shown in Figure 2-3. Radar reports enter continuously and asynchronously throughout each frame. Because of the smoothing and extrapolation methods chosen for the JSS tracking algorithm, the best and most recent estimates of track position are not available until the center of each frame, i.e., three seconds into the frame. Thus we cannot begin to correlate radar reports with the track data until half the frame is over. Furthermore, the tracking algorithms demand that all the coarse track-correlations be completed within 0.5 second after the end of the frame. At most, we have 3.5 seconds to perform coarse track-correlation--three seconds at the end of a given frame, and a half-second into the next frame.

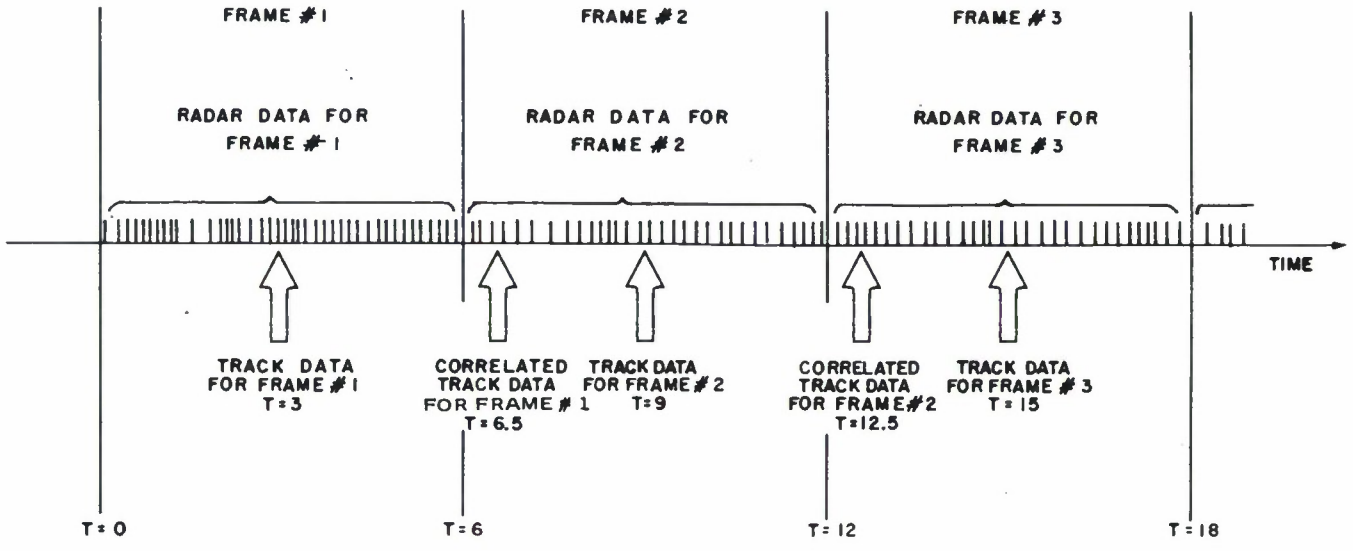


Figure 2-3 ORGANIZATION OF RADAR DATA INTO FRAMES

A very large portion of the computations performed at the input to the ROCC, in computers labelled A and B, consists of coordinate conversion and coarse track-correlation. If some means could be found to relieve these burdens, we might expect significant savings in computer hardware and a simplification of the remaining computer program. As discussed in the next section, a small special-purpose satellite processor can perform both functions at very low cost.

Additional functions could be accommodated by a more elaborate version of the satellite processor. Various options will be described in Section IV of this report. It should be noted that the cost-impact discussion in Section V will refer primarily to an "advanced" satellite processor which includes the mapping and masking options.

SECTION III

THE "BASELINE" SATELLITE PROCESSOR

Although there are many portions of the JSS problem that may offer attractive possibilities for cost reduction through the use of special-purpose satellite processors (e.g., in the generation of displays), this report will confine its attention to the front end of the ROCC computational burden, where the data from the surveillance radars are arriving at a high rate.

Since we intend to examine the cost benefits of satellite processors in a later section, we have too many design options at this point in our discussion. First, we have choices concerning the processing functions to be performed: Should the satellite perform mapping and masking as well as track correlation? Is it advisable to have the central computer worry about coordinate conversions? Could the satellite take over the entire tracking function, rather than just the coarse track-correlation? Second, we must decide on the burden to be placed on the JSS software programmer: To what extent should we expect the programmer to allocate core-memory space, do bookkeeping on data manipulated by the satellite, or concern himself with data transfers to and from the satellite? Third, we must use judgment in allocating hardware burdens to the satellite: How much internal memory should the satellite have? Is the satellite to be a fixed machine, or should it be programmable through the central computer? What provisions should we make here for input/output control?

What we shall do is define a "baseline" satellite processor, carry out a relatively detailed design for the baseline, and make cost estimates for the baseline. Then, in a later section of the report, we shall consider in less detail some of the more important design options.

The definition or list of requirements for our baseline satellite processor is presented in Table 3-1. These requirements will result in a special-purpose device which performs a reasonably complex set of tasks, which places a minimum burden on the hardware and software for the central computer, and which is likely to be relatively expensive (since minimum satellite cost is not a primary goal, and the other requirements will tend to maximize the hardware complexity of the device). As we shall see, the baseline satellite is still not very costly, despite these ground rules.

Basic Design Approach

Our general approach to the design of the satellite processor can be described with the aid of Figure 3-1. Radar data enter the central computer, at a maximum rate of 8500 reports per six seconds. Upon input, they are temporarily stored in buffer memory (not shown) and subjected to masking and mapping operations which limit the remaining data to a maximum rate of 3600 reports per six seconds. The central computer then sends each of the 3600 reports to the satellite processor, where they are stored in a RADAR REPORT TABLE. At this point, each report is represented by three 16-bit words, packed as follows:

Functions:

1. Coordinate conversion of all radar data not mapped or masked.

2. Coarse track-association: Each radar report will be "tagged" to show whether or not it lies within 12 NM of any track; the tag will not indicate which track(s) the report correlated with.

Programming Burden for the JSS Software: The satellite should be as simple to use as possible. The processor should require no bookkeeping by the programmer, and should not interrupt the central computer at an excessively high rate. The satellite should not require any significant increase in core memory in the central computer.

Hardware Tradeoffs in the Satellite: No attempt should be made to economize on the satellite's hardware at the expense of complicating either the interfaces or the JSS computer hardware/software.

Satellite Flexibility: The processor will be a fixed-program machine whose program is stored in a Read-Only-Memory (ROM). Site constants, however, will be read from the central computer whenever desired.

Input/Output: The satellite will use one standard bi-directional input/output channel of the central computer. Several control lines will be used to indicate status of the data passing over the single channel.

Table 3-1. "Baseline" Satellite Processor Requirements.

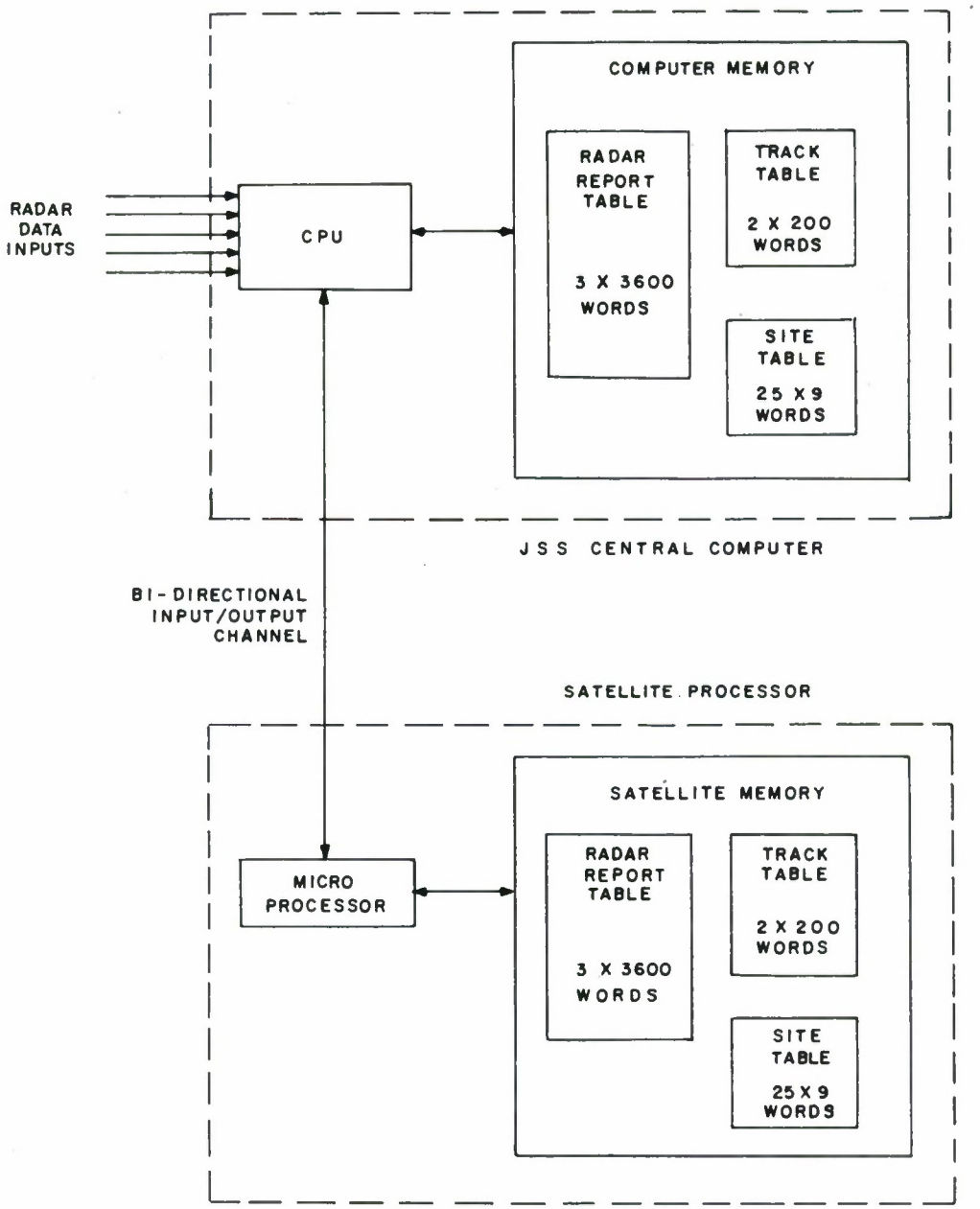
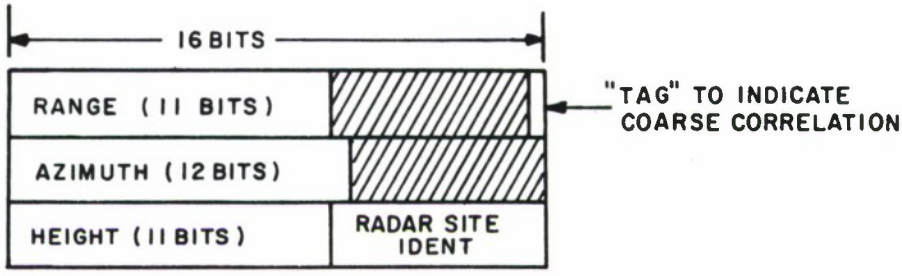


Figure 3-1 BASIC COMPUTER/SATELLITE CONFIGURATION



Assuming the bi-directional input/output channel to be a 16-bit data bus, there are at most 3 x 3600 16-bit radar-report words delivered to the satellite processor's memory every six seconds. These data arrive at the satellite more or less continuously throughout any given six-second frame.

About three seconds after each frame begins, the JSS programmer must arrange to send the appropriate track data to the satellite. It is assumed that the central computer maintains a TRACK TABLE which contains up to 200 pairs of X- and Y-coordinates, representing the estimated position of each track at the center of the corresponding frame; the table is updated every frame. Using the same input/output channel, the contents of the central TRACK TABLE are transferred to a similar TRACK TABLE in the satellite processor's memory.

At infrequent intervals, on demand, the nine constants K_1 through K_9 for each individual radar site are transferred over the channel and stored in the processor. In some cases the site may require more

frequent updating, because the JSS might include mobile (balloon-borne) radars.

At the beginning of each six-second frame, the raw radar reports for that frame begin flowing over the channel and are stored sequentially in the RADAR REPORT TABLE. The satellite's microprocessor immediately starts the execution of the coordinate-conversion equations; when it has completed the conversion for any given three-word report, it replaces the original R, θ, and H values with the corresponding values of X, Y, and H. During the first three seconds of the frame, the satellite can perform coordinate conversions, but cannot begin coarse track-correlation because the updated track coordinates are not yet available.

In the middle of the frame, the satellite's TRACK TABLE is loaded with fresh data from the central computer, and the microprocessor can begin to operate on the coordinate-converted data. It starts at the top of the RADAR REPORT TABLE, and loads a pair of X- and Y-coordinates into one of its internal registers. Then it sequentially tests that report against every pair of track parameters from the TRACK TABLE; if the vector distance between the report and any given track is less than 12 NM, the microprocessor adds a "tag" (e.g., sets the low-order bit) in the first word of the stored report to indicate coarse correlation between the report and the track. To complete the coarse-correlation process for the entire frame, the microprocessor must test up to 3600 radar reports against 200 target tracks.

Shortly after the end of the frame, the satellite has finished its coarse track-correlation tests. The RADAR REPORT TABLE will now contain up to 3600 three-word reports whose coordinates have been converted, and which have been "tagged" for coarse correlation. The entire RADAR REPORT TABLE is then transferred over the input/output channel and loaded sequentially into an identical table in the central computer's memory, to be used by the JSS programmer for display and final track correlation.

The various operations described above require two types of interleaving. First, the microprocessor must work its way down the RADAR REPORT TABLE, executing the coordinate-conversion equations for each report; halfway through the frame, it must simultaneously perform coarse track-correlations beginning with the top of the table. Several internal counters are needed to remember how many raw reports have been received from the central computer, how many have had their coordinates converted, and how many have been tested for coarse track-correlation. Second, the input/output channel must be shared by four types of data: site constants (sent infrequently to the satellite), raw radar reports (sent continuously to the satellite on a one-at-a-time basis), track data (sent quickly in a single block to the satellite), and processed radar data (sent quickly in a single block back to the central computer).

JSS Programming Considerations

From the standpoint of the JSS software programmer, the satellite can be considered almost as a subroutine. The programmer must remember

to do a few things in the correct order, but these should not cause him much trouble.

First, he must arrange to send over the site constants as frequently as the radar situation requires; this might be done once per day for fixed sites. The programmer loads a single table with all the constants for all the sites (nine constants for up to 25 radars), sends a signal to the satellite indicating that site data are on their way, and executes a Direct Memory Access (DMA) transfer of the entire table to the satellite.

Next, he must send the raw radar reports to the satellite. Since he must already have internal buffer tables to accomplish the mapping and masking operations, this transfer is merely a matter of reading the appropriate data from the tables in sequential order. The radar reports in the central computer contain some data of no interest to the satellite--e.g., the time associated with the radar detection, or various status indicators--and the programmer must extract only the range, azimuth, height, and site identification and pack them into three sequential 16-bit words. Once packed, he may send the words in blocks or one at a time, and it is not necessary for him to count the number of reports he has sent. Whenever he sends a radar report, he must first set a control line which allows the satellite to interpret the incoming data as radar information.

During the collection of radar data, some other portion of the central computing facility is preparing the latest estimate of track positions for each target identified by the system. These estimates

must be loaded into a single TRACK TABLE, and the entire table must be transferred to the satellite at about the middle of the frame. The programmer must set an appropriate control line to inform the satellite.

Finally, shortly after the beginning of each frame, the JSS programmer must request a DMA transfer of the processed data from the satellite into whatever table locations he wishes in the central computer's memory. The programmer can rely on the completion of the transfer before 0.5 second into the frame, or he may test a control line from the satellite.

The satellite requires a "start of frame" signal from the central computer; if this is not provided by the system hardware, it may be necessary for the programmer to set a control line at the proper time.

In some computers (e.g., the larger IBM computers), a DMA transfer of data does not interfere with the operation of the central processing unit after the transfer has begun. In most other machines, each item of data transferred over the input/output channel "steals" a cycle from the central processor, and slows it down. DMA transfers at the highest data rate can therefore stop the entire computer for the duration of the data block. Since the complete shutdown of the computer for DMA transfers to and from the satellite processor would represent an awkward constraint for the JSS programmer, the satellite timing has been designed to ensure relatively slow DMA transfers. Consequently, the programmer need not worry about such transfers, and can execute them any time he wishes.

Computations Within the Satellite Processor

Since the baseline processor is expected to perform a rather wide variety of computations, it has been configured as a general-purpose computer with some auxiliary special-purpose hardware. The basic computational arrangement is shown in Figure 3-2.

The heart of the satellite is a 16-bit microprocessor. For the baseline design, a relatively fast computer is needed, and we have chosen to implement the CPU in the form of four four-bit "slices" which can operate at a clock rate of 5 MHz if desired*. Program instructions are 24 bits long (containing 9-bit micro-instructions), and the computer can manipulate 16-bit data words. Although the basic computer can add, subtract, shift, and perform many logical operations, it does not have hard-wired multiply or divide instructions; these must be implemented as subroutines. The microprocessor includes 16 general-purpose registers which may be used for indirect addressing, and which can be incremented or decremented automatically as part of some other operation such as fetching data from memory.

The most sophisticated computations required of the satellite involve the evaluation of the coordinate-transformation equations shown in Section II. The transformations call for the calculation of two sines, two cosines, a square root, and several multiplications and additions. All arithmetic is performed in fixed point. As described

*The specific element is the AM2901 bipolar four-bit slice processor manufactured by Advanced Micro Devices, Inc.

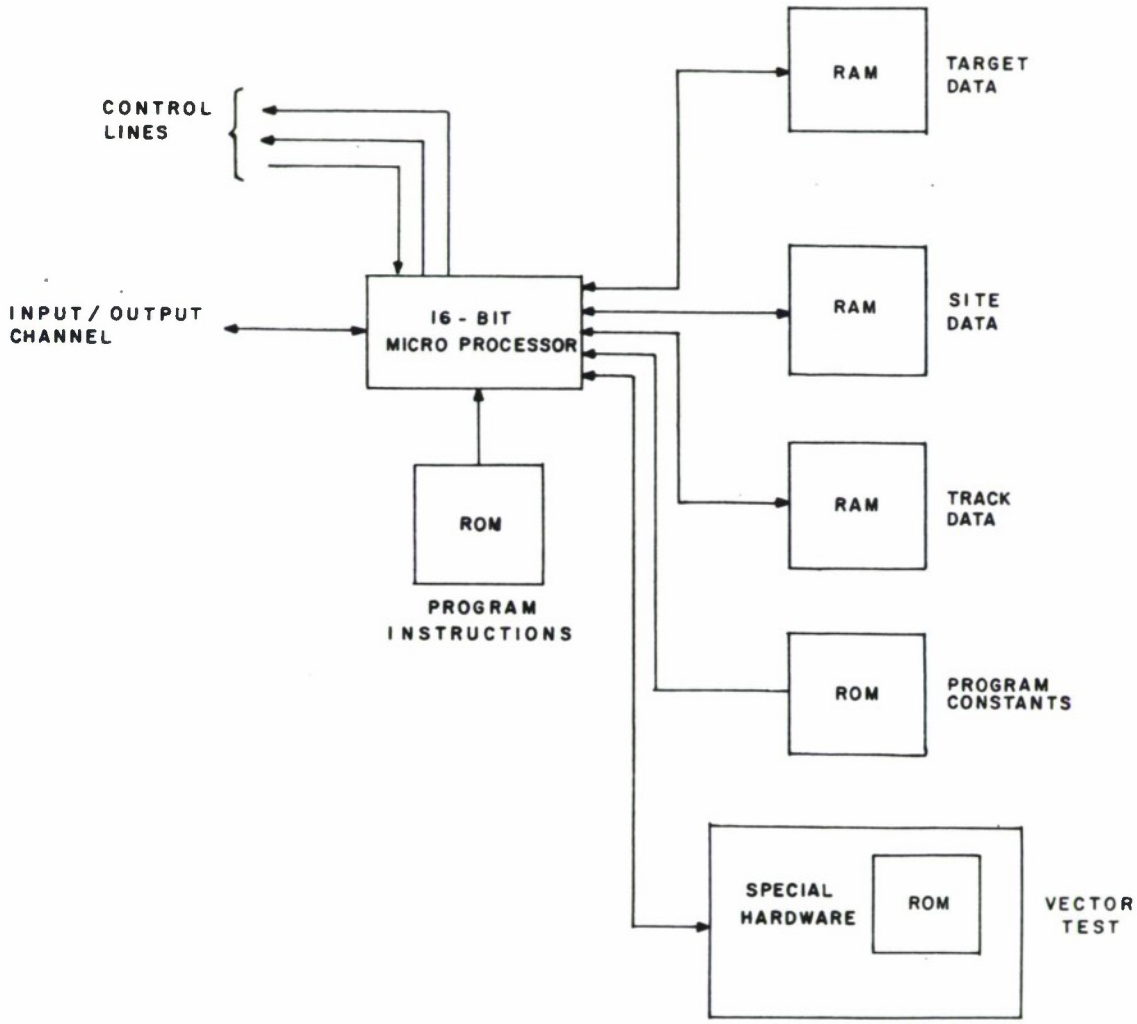


Figure 3-2 COMPUTATIONAL ARRANGEMENT FOR BASELINE SATELLITE PROCESSOR

in Appendix A, four subroutines are used for multiplication, sine, cosine, and square root. It is estimated that the total coordinate-conversion portion of the program consumes 124 instructions and uses 36 stored constants. Since the microprocessor handles instructions and constants (data) differently, they are stored in two different read-only memories.

The second type of computation involves the tests for coarse track-correlation. To test a given radar report against the 200 tracks, the microprocessor first fetches the X- and Y-coordinates of the radar report and loads them into general-purpose registers. Then it begins testing against the stored track data. In one instruction it fetches the X-coordinate of the track and subtracts it from the X-value of the radar report; in the next instruction it sends the resulting value of ΔX to the Vector Test Unit shown in Figure 3-2. In the following two instructions it fetches, subtracts, and delivers ΔY to the Vector Test Unit. Since the fetch instructions automatically increment the fetch-address registers, only six computer cycles are required to produce ΔX and ΔY (two are need for counting and jumping).

The Vector Test Unit is a simple special-purpose device containing some logic elements and a read-only memory. Its function is to rapidly implement the following test:

$$T = \sqrt{(\Delta X)^2 + (\Delta Y)^2}$$

If $T \leq 12$ NM, then the radar report is considered to be correlated with the track.

The logic elements merely examine the high-order bits of ΔX; if any of these bits are set, then T will clearly fail the test. Similarly, the high-order bits of ΔY are tested. If neither ΔX nor ΔY are so large as to fail this crude test, then ΔX and ΔY are each used as five-bit addresses for a read-only memory. For any given pair of ΔX and ΔY values, the memory contains either a zero or a one, indicating the failing or passing of the coarse track-correlation test. Thus the 12-NM vector test can be hard-wired with a relatively small number of digital logic chips, and this burden can be removed from the microprocessor.

The microprocessor must perform many other functions, each of which is almost trivial. As examples, it must participate in the storing and reading of data transferred during DMA interchanges, and must maintain eight of its general-purpose registers for keeping the last radar data address entered, the current track data address, the last track data address, etc. The details of the microprocessor program needed to perform these miscellaneous operations have not been worked out, but there appears to be more than adequate capacity in the 256-step program memory provided in the baseline design: If the coordinate-conversion program uses 124 steps, and the coarse track-correlation uses about 20 steps, then about 112 steps are left for miscellaneous tasks.

Satellite Timing Estimates

As discussed in Appendix A, it is estimated that the coordinate conversion of one raw radar report will require about 800 machine cycles.

If we operate the satellite-processor clock at 3 MHz (a value comfortably below the manufacturer's specified maximum clock rate of 5 MHz), then one conversion will consume about 267 microseconds. For a maximum of 3600 radar reports per frame, coordinate conversion will require a total of

3600 x 267 microseconds = 0.96 seconds.

Appendix A also estimates that the attempt to correlate one radar report with up to 200 tracks will require about 1200 machine cycles, or 400 microseconds. At the upper limit of 3600 reports, the total time required for coarse track-correlation is 1.44 seconds.

Ignoring site data, we must transfer

3 x 3600 16-bit words to the satellite (radar reports)
3 x 3600 16-bit words from the satellite ("tagged" radar reports)
2 x 200 16-bit words to the satellite (track data)

or a total of 22,000 words every six seconds. If we assume that the transfer of each word will interrupt the microprocessor for three machine cyles, then the total time expended in transfers of data is 22 milliseconds. Thus data transfers impose a negligible burden on the microprocessor (and the central computer as well).

As shown in Figure 3-3, the process of coordinate transformation can be carried out through most of the six-second frame, but the coarse track-correlations must be done only during the last three seconds of the frame; therefore, the second half of the frame is the busiest. According to the estimates given above, the microprocessor will take

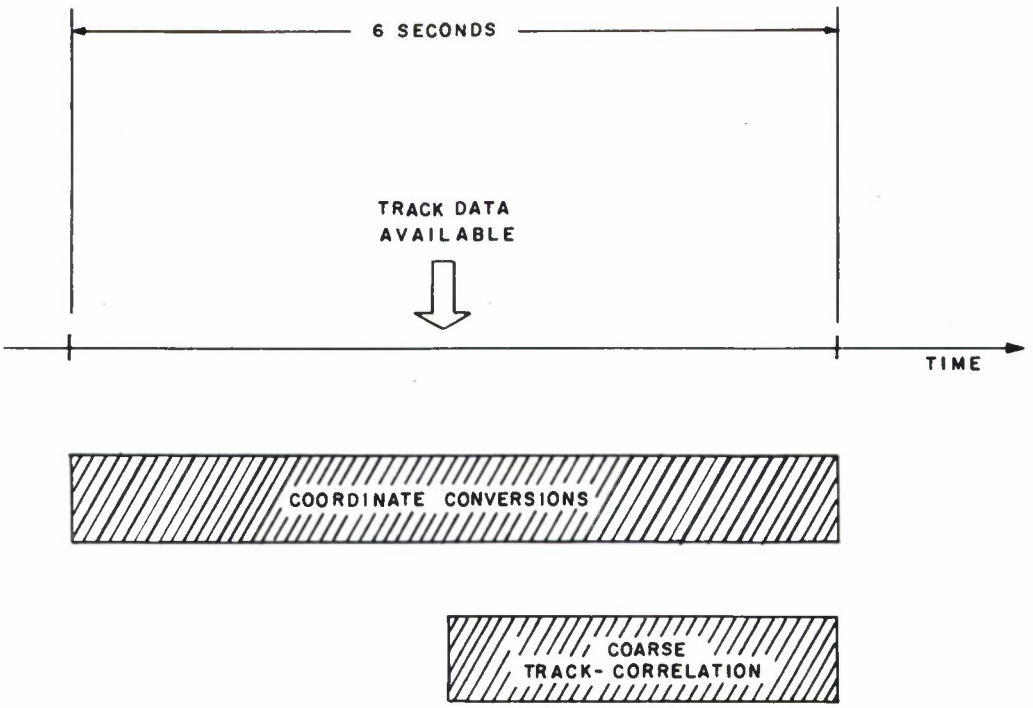


Figure 3-3 TIMING OF TASKS WITHIN DATA FRAME

1.44 seconds to perform all the coarse track-correlations, and 0.96/2 seconds to do half of the coordinate conversions, for a total of about 1.92 seconds of CPU capacity, leaving another 1.08 seconds for miscellaneous and overhead functions during the last half of each six-second frame. These estimates assume that the radar data entering the system are uniformly distributed in time, an assumption which should be nearly correct when many independent radars are providing the data. In the worse case, where all of the data in a frame enter during the last three seconds, the microprocessor will take 1.96 seconds for correlation and 0.96 second for coordinate conversion, or a total of 2.40 seconds (leaving 0.6 second for miscellaneous chores). We may conclude that a single microprocessor can accommodate the satellite's needs, even for the most unfavorable distribution of raw radar data.

Overall Design of Baseline Satellite Processor

Having discussed the various major items in the processor's design, we may now consider the overall block diagrams of the device. Figure 3-4 shows the microprocessor and its various memories, and Figure 3-5 shows the ten control functions connecting the central computer to the satellite.

The four microprocessor "slices" do not actually constitute a computer, but must be supplemented by a look-ahead "fast carry" chip that services the four CPU slices, two program-sequencer chips, three read-only memories containing 256 24-bit program instructions, and a 24-bit latch to hold the last program instruction. Hence, we need eleven chips to form the computer.

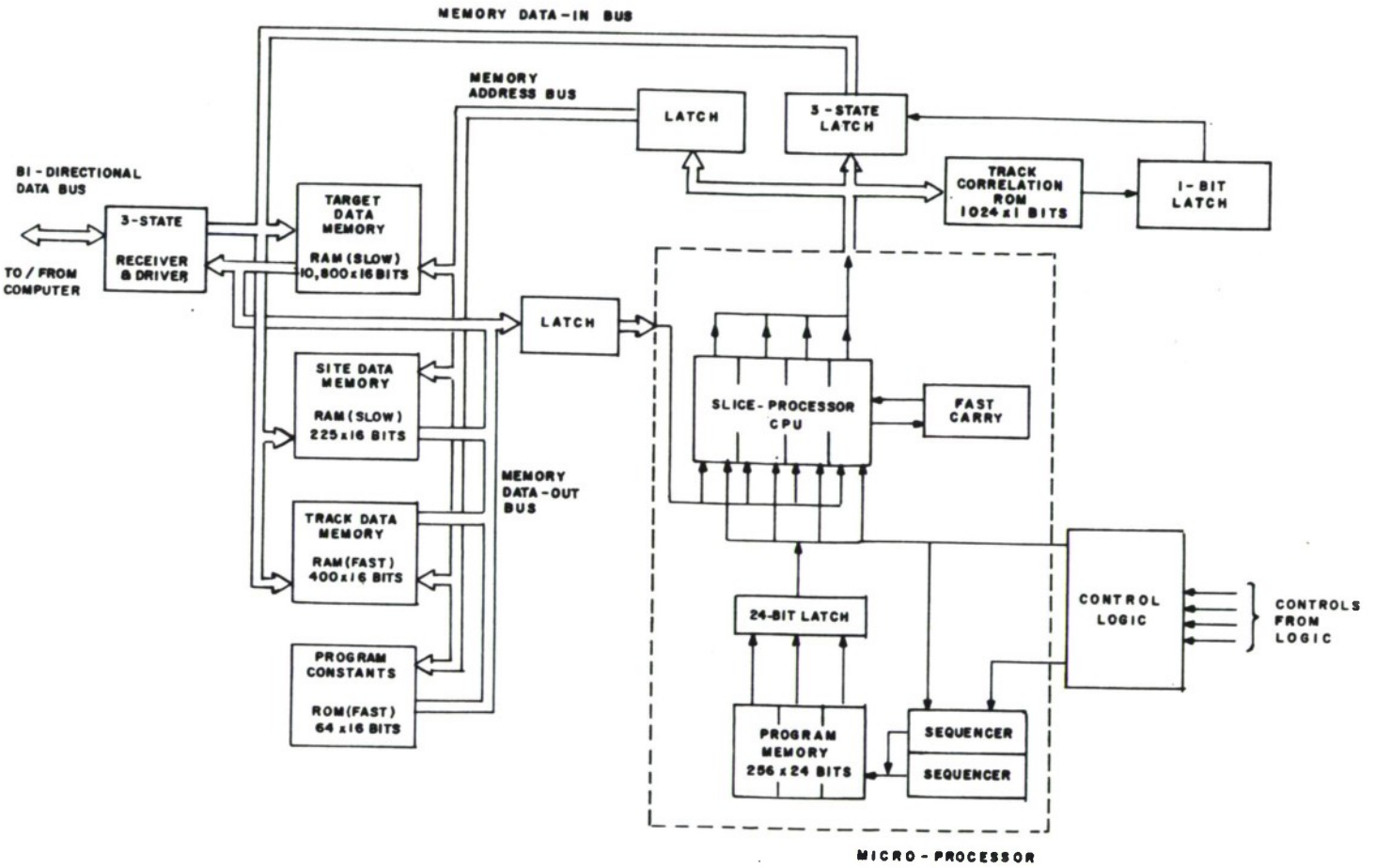


Figure 3-4 BLOCK DIAGRAM OF SATELLITE PROCESSOR

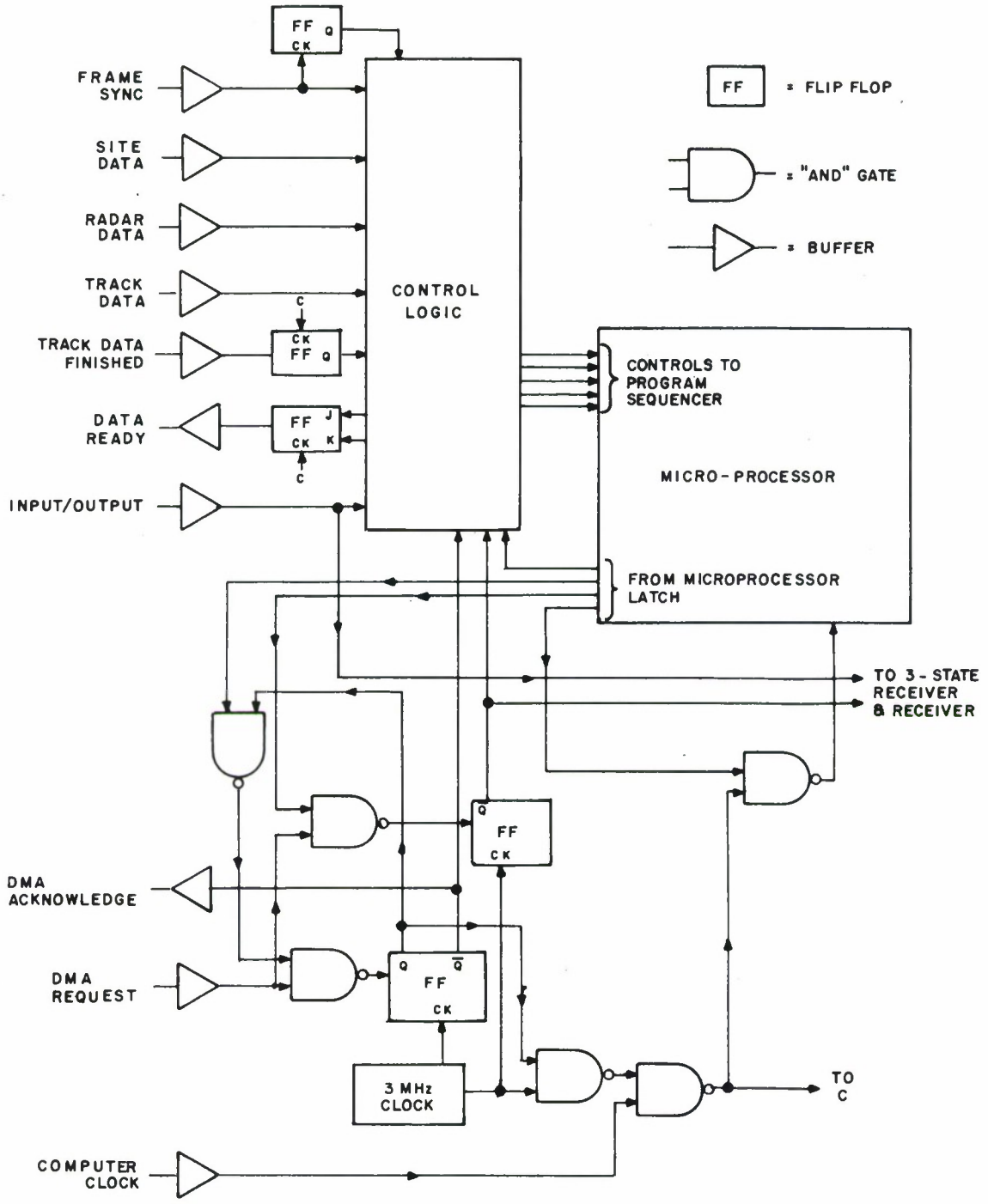


Figure 3-5 CONTROL LINES FOR SATELLITE PROCESSOR

A large fraction of the satellite's hardware (79 LSI packages or chips out of a total of 111) is devoted to digital memories. The random-access memory used to store the radar reports requires 50 LSI chips capable of holding over 170,000 bits; however, the organization of the satellite permits this memory to be relatively slow. The present line of commercial slow random-access memories offers 4096 bits of storage in a single chip; improved memories giving 16,384 bits on one chip will be available soon, promising a dramatic reduction in the size and cost of the overall unit. Sixteen integrated-circuit packages are required for the fast random-access memory used to hold the 200 pairs of track coordinates.

The ten control functions between the satellite and the central computer are used as follows:

1. Computer clock to satellite.
2. DMA request to satellite.
3. DMA acknowledge to computer.
4. Beginning-of-frame signal to satellite.
5. Signal from computer to indicate whether the input/output channel is being used as input or output.
6. Signal to the central computer indicating that the processed data are ready for transfer back to the computer.
7. Signal to satellite indicating the end of the track data.
8. Site data
9. Track data
10. Radar data

} Signals from the computer indicating the type of data being sent over the channel.

Obviously, these ten control functions can be encoded to reduce the number of physical control lines between the computer and the satellite, or to accommodate computer constraints. Control function No. 6 can be eliminated if the JSS programmer is willing to wait the full 0.5 second after the start of a frame to begin his manipulations of the processed data.

These control functions permit a very flexible approach to data transfer, and relax many constraints on the JSS programmer that might ordinarily be imposed. For example, control function No. 7 (end of track data) allows the programmer to send the track data either in a block or as a separate set of transfers; furthermore, he can finish his track transfer at any reasonable but arbitrary time, and need not send the entire maximum list of 200 track coordinates if less than 200 are given. It is assumed that the JSS programmer will send over the entire list of 9 x 25 site constants whenever the site data are to be changed.

As mentioned above, the microprocessor uses eight of its internal general-purpose registers as address registers to control the various interleaved sequences of coordinate-conversion and coarse track-correlation computations. Another problem to be accommodated is the overlap in the processing for successive frames; when data from the previous frame are being returned to the central computer, new radar data are being sent to the satellite. Appendix B contains a somewhat more detailed description of the use of the reference address registers, as well as comments on DMA operation.

The block labelled "logic box" in Figure 3-5 is implemented with a programmed logic array plus associated small-scale integrated components. Its purpose is to generate the appropriate program step or input/output control signals corresponding to the state of the program and the input/output control lines. It causes the program sequencer to increment until a program "jump" condition or a DMA requirement is encountered. When responding to a DMA request, the request is not relayed to the control logic until those times in the program when it is convenient for the microprocessor to react, i.e., generally not during a subroutine. A flip-flop then transfers this request to the control logic which acts according to the status on the input control lines in directing the sequencer to the appropriate program step. When this occurs (at the internal clock rate), the satellite processor issues a DMA acknowledge signal and accepts the clock signal from the computer. The DMA condition continues until the DMA Request control line is returned to normal, at which time the normal operation of the processor resumes.

Hardware Cost Estimates for the Baseline Processor

As discussed previously, the satellite baseline processor could be fabricated using approximately 111 integrated-circuit packages. Of these, 11 are associated with the implementation of a general-purpose microcomputer, 79 are various types of random-access and read-only digital memories, and the remainder are miscellaneous logic circuits.

Although we have considered all of the essential elements and many of the details of the baseline processor, we have not attempted

to carry out a fully complete design. Our primary purpose in this study is to estimate the characteristics and impacts of such a processor, and a totally finished design is not necessary to meet these goals. In addition, the JSS itself has not been specified in any great detail at the present time, and even the types of computers have not been selected; the principal uncertainty in our satellite design lies in the input/output and interface considerations, and they cannot be addressed explicitly until the central computer has been chosen.

Thus, the estimates of the cost of the satellite processor are not exactly correct, but they are believed to be accurate to within a few hundred dollars.

Table 3-2 presents a list of integrated-circuit components, specified by manufacturer's part number except for some minor items, for the baseline processor. The costs are given for unit quantities, and take no advantage of volume purchase. (It should be noted that a 10 to 30 percent discount could be obtained if the 111 LSI packages for just one satellite were bought through a single vendor.)

The cost of assembling these components onto boards, installing them into a suitable box with the necessary connectors, switches, lights, power supplies, etc., and a minimum of testing, can be estimated on the basis of between six and ten dollars per integrated circuit. Six dollars per integrated circuit is often quoted in industry for this cost; we will use eight dollars to be conservative. The total cost for assembly, installation, and testing is therefore estimated to be $890.

MICROPROCESSOR:					
CPU	AM2901	4 ea	@	$ 60.00	$ 240
Sequencer	AM2909	2 ea	@	42.12	84
Fast Carry	AM2902	1 ea	@	5.67	6
Latch	N74S174	4 ea	@	7.69	31
ROMS:					
Program	SIG 8204	3 ea	@	21.00	63
Constants	82S123	4 ea	@	6.45	26
Correlation	82S123	2 ea	@	6.45	13
RAMS:					
Target Data	SIG 8107B-4	50 ea	@	16.00	800
Track Data	82S11	16 ea	@	45.00	720
Site Data	AM2971	4 ea	@	20.00	80
ASSOC. MSI LOGIC:		10 ea			100
I/O CIRCUITS:		10 ea			100
OSCILLATOR:		1 ea			15
		111 packages			$ 2277

Lumped estimates are given above for miscellaneous MSI digital logic and for the input/output circuits.

Table 3-2. Cost Estimates for Integrated-Circuit Components.

The total cost for components and construction of the baseline satellite processor is estimated as follows:

Component Cost	$ 2277
Assembly Cost	890
Total	$ 3167

The design and development costs to take the baseline from its present status to a completely operational prototype should not take more than a few weeks of engineering time, once the detailed characteristics of the central computer interface have been specified.

The baseline processor has not, of course, actually been fabricated. We can approximate its physical form by mounting 111 integrated circuits on a standard printed-circuit board measuring 17 x 8 inches. Figure 3-6 is a photograph of such a simulated device. In the JSS environment, it could be incorporated in other digital devices having spare slots (e.g., the central computer) or a separate box with power supplies could be devised.

Programming and Testing the Satellite Processor

In some applications, the programming of a microprocessor can represent a costly four-fold chore--the basic program is first developed and tested on a large computer using a high-order language such as FORTRAN; the symbolic code is next converted to microprocessor machine instructions using a "cross-compiler"; the machine-language program is then tested on an "emulator"; and finally the program is loaded into a PROM, and the whole assembly tested in actual operation. It can be argued that this

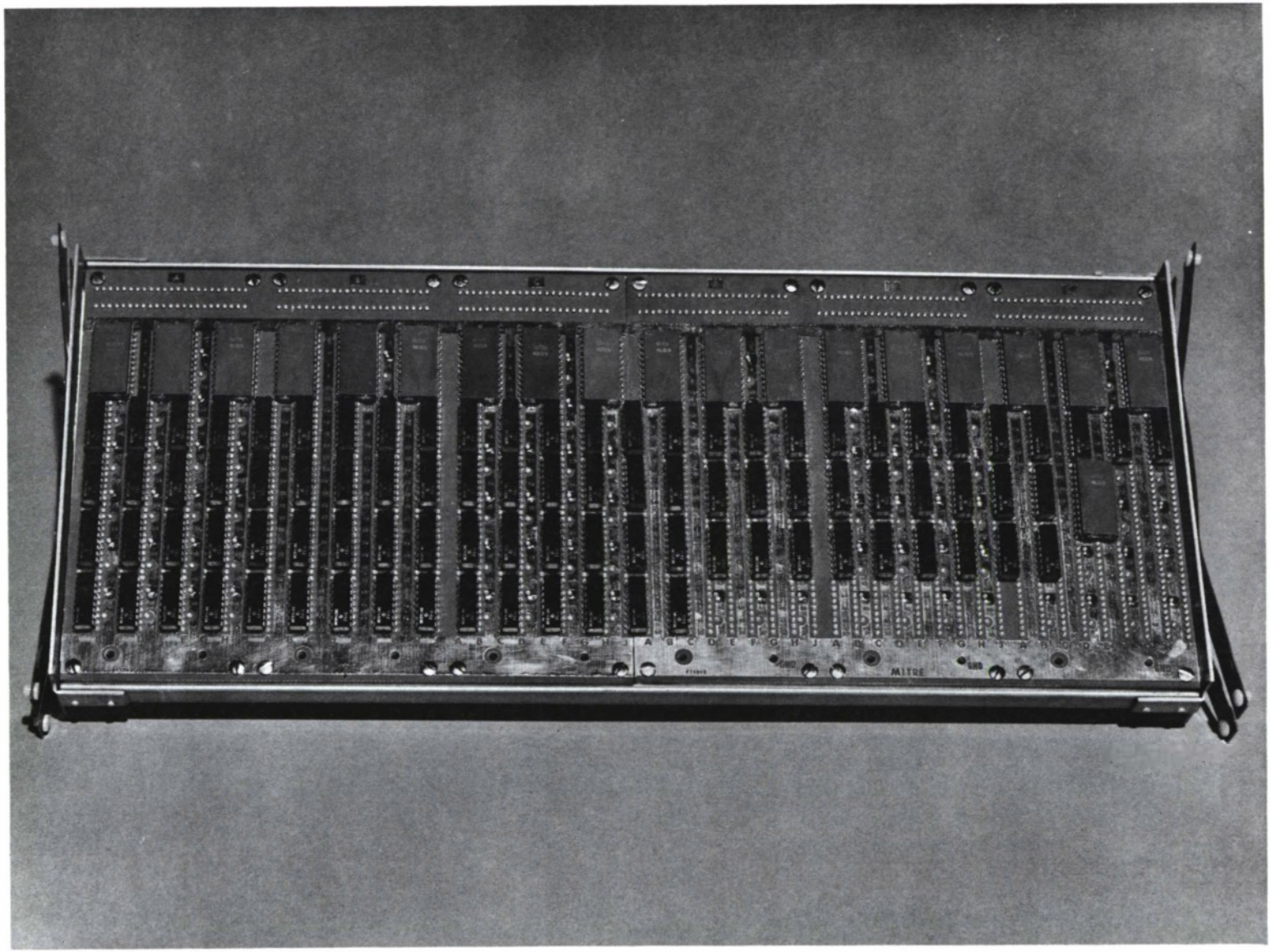


FIGURE 3-6 Photograph of Simulated Processor

process is much more expensive per finished instruction than the equivalent operation on a conventional large-scale computer.

The program for the satellite can be developed without most of these time-consuming steps, because the basic processor functions are so simple. By using certain efficient testing techniques, the entire procedure can be completed in a few man-weeks at a correspondingly low cost.

As described earlier, the complete program occupies less than 256 machine-language instructions. A program of this size can be written directly in machine language by one knowledgeable person in a day or two, on the basis of the brief programming analyses presented here and in Appendix A. It should not be necessary to deal with higher-order languages, or even an assembly language (although the latter would be convenient).

Testing of the satellite processor and its program can proceed simultaneously, in two distinct phases: laboratory or "bench" tests, and operational tests using the central computer as the principal tool for exploration. These two phases and the test methods employed in each are outlined below. There should be no need to utilize an "emulator" or any similar indirect techniques.

We assume that the satellite processor is physically complete, and that the operational program has been written and loaded into the PROM associated with the satellite's microprocessor. We also assume that the laboratory has the standard equipment to load PROMs, using paper-tape

input; the procedure is to take a hand-written assembly-language program, punch a paper-tape version of the program, load the program into the PROM, and physically insert the PROM into the satellite.

In the laboratory testing phase, the traditional approaches to debugging digital hardware can first be employed on the satellite processor, to assure ourselves that the presence of certain signals on the control leads will cause the desired logical actions within the unit. Next, some embryonic test programs (consisting perhaps of 10 or 20 machine-language instructions) are loaded into the PROM, to cause the microprocessor to transfer data from one portion of the satellite's memory to another; these transfers will be performed repetitively, and the movement and arrival of stored data can be observed with an oscilloscope. When these simple operations can be executed without error, the full program is loaded into the satellite, and a few of the control lines (e.g., the "frame sync" line) can be pulsed with external test equipment at the proper rates. The satellite should cycle through its program, and exhibit specific actions at certain times within the cycle, thereby giving some general indication that the unit is functioning as expected. The principal purpose of the laboratory phase is discover any hardware-oriented errors.

In the second or operational phase of testing, the satellite is connected into the central computer using all of the interfaces planned for full-scale operation. Then a series of evolutionary tests are arranged, where each test calls for the writing of a simple program

(in high-order language) for the central computer, and either a short assembly-language program or the full-scale program for the satellite.

An example set of tests might proceed as follows. The central computer would first be programmed to send the "track" table to the satellite once per frame, and the satellite would be programmed to receive the "track" table and store it in its corresponding RAM; it should be possible to write and load these two short test programs in less than one hour. In execution, the behavior of the control lines and the interfaces, and the movement of the data from computer to satellite can be observed directly with the proper test equipment, and any problems should be simple to diagnose. Similar separate tests can verify the performance of the commands to move target data and site data to the satellite, and tagged target data from the satellite to the computer. Next, a series of tests of gradually increasing complexity can be used to test the overlapped transfers of data back and forth, the coordinate-conversion portion of the processing, the track-correlation portion, and finally the combined operation of all processes.

It is expected that ten to fifteen short test programs would be written during the second phase; and, if the operational satellite program initially contains several errors (which is likely), the full program will have to be laboriously re-written many times before it is completely satisfactory. It is important to note, however, that the test programs are individually trivial, and the full operational program can be modified in a few hours.

Using the methods outlined above, we believe that a single person could comfortably program and test the baseline satellite processor in a few weeks. This person would have to be familiar with the central computer's software, as well as knowledgeable in both the hardware and software aspects of the satellite's microprocessor and its interfaces.

SECTION IV

OTHER SATELLITE PROCESSOR OPTIONS

The previous section has described a "baseline" satellite processor for the JSS, designed to perform the functions of coordinate conversion and coarse track-correlation. Both the functions and the design of the baseline device were selected rather arbitrarily, for the purpose of presenting a specific example in enough detail to make the conclusions (with regard to programming and cost) credible. In this section we will discuss a series of design options for a JSS satellite processor. We will not attempt to consider all of the possible options, nor to give detailed information about any given version; instead, we shall merely outline the more important characteristics of some options, and rely on the reader to make the necessary extrapolations in complexity and cost with respect to the baseline satellite.

First we shall discuss a few variations on processors which perform essentially the same functions as the baseline satellite. Then we shall suggest other additional functions that might be performed by a suitable processor. As before, we shall confine ourselves to those operations near the front end of the JSS.

Minor Variations on the Baseline Satellite Processor

The baseline processor could be implemented in the form of two separate processors, one devoted to coordinate transformations,

and the other dedicated to coarse track-correlation. Presumably each device would employ its own micro-processor and would incorporate its own internal memory; each would need a bi-directional interface with the central computer. Although the equipment would be more costly than the baseline device, the two units would have greater computational capacity, and therefore could handle a larger data load if desired.

A simpler satellite could be devised which relied on the central computer for storage of the raw radar data; memory requirements in the satellite would thus be greatly reduced. In this version, the satellite would sit idle until the track data could be transferred from the central computer (about three seconds after the start of the frame). Then each raw radar detection would be sent to the satellite, processed, and returned to the computer on a one-at-a-time basis; there would be no need for the satellite to "remember" more than one radar detection. The penalty for this simplicity is a small amount of additional difficulty in using the satellite. It would be necessary for the JSS programmer to arrange the data transfers with more care, and to perform some minor bookkeeping on the number of radar data points. A secondary variation on this one-at-a-time technique is to utilize a small buffer memory for radar data in the satellite (capable of storing, say, 100 detections); then the data transfers could be performed more easily by the JSS programmer, but he would have to expend somewhat more care in the bookkeeping portion of his responsibilities.

Increasing the Capability of the Satellite Processor

Several embellishments on the basic baseline processor are discussed below.

Additional Correlation Data

The baseline processor accomplishes the coarse track-correlation task by adding a "tag" to the radar data; presence of the tag indicates that the radar detection was within 12 NM of at least one of the target tracks. With a relatively small addition to the satellite's hardware, it would be possible to give the JSS central computer more information about the correlations. In particular, the satellite could identify which track(s) correlated with each radar detection, and could also give the vector distance between the detection and the track. If the tracking algorithms used in the JSS operate on the basis that no more than one radar detection should be associated with each track, and/or that each track should be associated with no more than one detection, then the satellite could perform these decisions and selections prior to sending the tagged data back to the central computer. These burdens could be rather easily accommodated by a satellite processor, and could lift a considerable computational load from the JSS computer.

Mapping and Masking

One of the important functions of the central computer is to limit the amount of incoming data, so that subsequent processing will not be overtaxed. When the input data loads are too high, the JSS rejects certain data on the basis of masking and mapping operations,

which are sets of tests dependent upon the geographical position and the type of each radar detection. The raw radar data can enter the system at a total rate as high as 8500 detections in any given six-second period; the mapping and masking procedures must place an absolute upper bound of 3600 detections in any given six-second frame. It is certainly feasible for a satellite processor to perform these operations, and thereby relieve the central computer from any direct interaction with the raw radar data. A separate satellite could be designed to handle the mapping and masking tasks, or these functions could be incorporated into the same device which executes coordinate conversion and coarse track-correlation on the radar data. Since all of these operations require the shuttling back and forth of fairly large quantities of radar data, it seems more efficient (from the JSS programmer's point of view) to combine all of the functions into one device.

The decision rules upon which mapping and masking are implemented are fairly complex, and will undoubtedly be revised in accordance with practical field experience. Generally, they involve two main elements. First, geographical regions are defined, and each data point is tested to determine whether it lies within the given region; each region is described in terms of two ranges and two azimuths, and its location and size can be adjusted by the JSS operators. Typically there will be many such regions, in areas of the total JSS coverage where target traffic may be heavy but not particularly

interesting (e.g., near an airport). Second, the tests are dependent in a complicated way upon the type of radar data under examination. For example, data may be identified as a "skin" radar report, an IFF report, a radar detection with or without height information, or one of a class of special synthetic test targets.

None of the operations described above are difficult to execute, and are quite appropriate for a fixed-point arithmetic unit of modest word size--i.e., they are well suited to implementation on a microprocessor. However, the decision rules are likely to change during the development and exercise of the JSS, and it would not be desirable to produce a satellite processor whose internal software could not be changed easily. One solution to this problem of software flexibility is to include the facility for storing the satellite's program in the central computer, and reading that program into the satellite's temporary program storage from time to time. In this fashion, the JSS programmer would have complete control over the particular decision algorithms being executed in the satellite, and could change them whenever he wished.

Unpacking

The raw radar data entering the JSS contain information that is not directly of interest to the baseline satellite processor. For example, the data include the time at which the detection or report was made, and various status indicators. For the baseline design, the JSS programmer was expected to sort out and "unpack" the raw data,

and rearrange only the pertinent information into a particular format (e.g., three 16-bit words). Later, after these data had been processed by the satellite and returned to the central computer, the JSS programmer was expected to associate each processed report with its corresponding auxiliary information. Although these JSS programmer responsibilities are very simple, even they can be eliminated by a suitable change to the baseline satellite design.

The satellite could be configured to accept truly "raw" data from the radar sites, without requiring any changes in format or any attention whatsoever from the JSS programmer. The satellite could take responsibility for sorting out all the auxiliary information, re-formatting internally where desirable, and could then transfer all the data back into the central computer's memory after the mapping, masking, coordinate conversions, and coarse track-correlations had been completed.

SECTION V

COST IMPACT ON THE JSS SYSTEM

Our principal motive in studying "satellite" processors as applied to JSS has been a significant reduction in system costs. It is unfortunately true that, while specific hardware designs for special-purpose devices can be developed easily, it is extremely difficult to assess the quantitative savings in cost which these devices might allow.

We begin by noting that the cost of the baseline satellite processor is negligible in comparison with the cost of the entire JSS ($5000 component cost plus a relatively small development cost for the satellite, versus an estimated cost of several millions for JSS equipment, software, and integration). Other more elaborate versions of satellite processors which include, for example, the ability to perform masking and mapping will require more components and involve more development; nevertheless, the total cost of even an advanced satellite can be neglected.

There are many contributors to the overall cost of a large system: hardware, software, testing, documentation, maintenance, training, spares, etc. It is well beyond the scope of this report to consider any but the first two items. We can look for hardware savings from the satellite-processor, in terms of reducing the number of computers in the system or through a reduction in the size of the computer(s). We might anticipate software savings, because the use of the satellite-processor would "free up" a relatively large portion of one or more computers, thereby relaxing some of the constraints on optimization of

real-time programs. Or we could argue for both hardware and software savings simultaneously. As we shall see, it will be difficult enough to quantify savings for either hardware or software alone, and we can only speculate on the magnitude of any combined advantages.

Let us begin by considering the possibility of hardware savings. Our task would be made immensely easier if the initial system configuration (before the introduction of a satellite processor) has already been subdivided into relatively independent computers; if we can eliminate one or more of these computers by adding a satellite, then the hardware savings can be directly attributed to the processor. In the case of JSS this process is blurred because the configuration of the central computing facility for the various ROCC's has not yet been finalized, and we are forced to employ our own judgement or the informal estimates of industrial contractors.

Rather early in the history of the JSS program, MITRE generated several different configuration concepts in the course of its own internal analyses. One of these configurations was shown in Figure 2-2; it has no special merit, but at the time it did seem to be one reasonable approach to the design of a ROCC. Each of the computers shown in Figure 2-2 is a "medium scale" computer, roughly in the same class with the Hughes H5118M general-purpose digital computer.

The H5118M computer is a modular, 18-bit, binary, parallel, synchronous system intended for real-time command and control applications. It can execute a fixed-point addition in 2.0 microseconds and a fixed-point multiplication in 4.6 microseconds, and its memory can be expanded

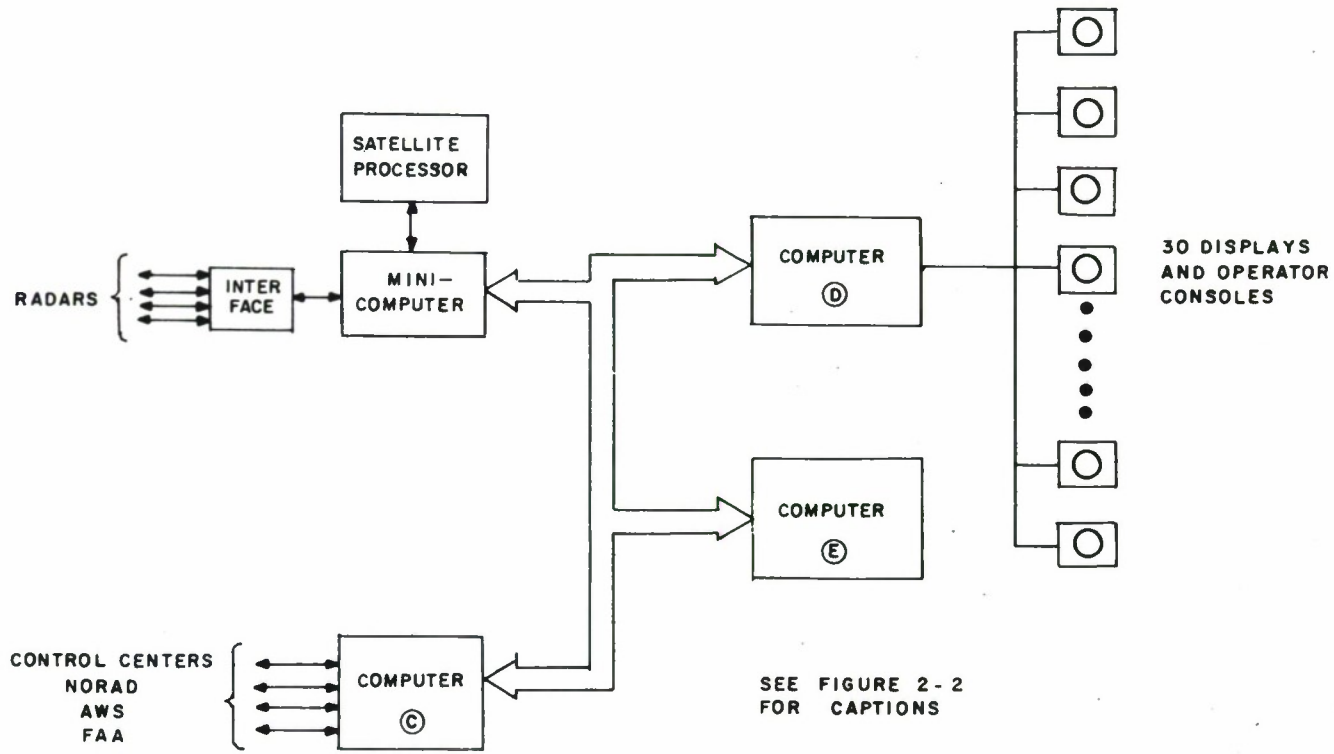


Figure 5-1 SUGGESTED JSS CONFIGURATION USING SATELLITE PROCESSOR

to a maximum of 131,000 words. The cost for an H5118M with dual processors and full memory lies between $300,000 and $400,000. On this basis, the total cost for the computers shown in Figure 2-2 is nearly two million dollars, not including the additional computers required for redundancy and testing.

If the baseline satellite processor (which performs coordinate conversion and coarse-track correlation) were used with this system configuration, then one of the two computers at the input end could be eliminated. The remaining input computer would handle the multiple-radar data, perform the masking and mapping operations, communicate with the satellite processor, and interface with the rest of the system. Thus the hardware savings could be greater than $300,000.

If an "advanced" satellite processor, capable of performing the masking and mapping operations as well as coordinate conversion and coarse-track correlation, were used then both medium-scale input computers could be eliminated and replaced by a single mini-computer as shown in Figure 5-1. The mini-computer's functions would be merely to input the raw data from the multiple radars, communicate with the satellite processor, and interface with the remainder of the system. The hardware savings could be greater than $600,000 in this case.

The previous discussion was based on the initial configuration shown in Figure 2-2, with the assumption that both of the input computers were of the "medium scale" class. Later studies by MITRE and industrial contractors have suggested that a somewhat smaller set of computers

might suffice for the ROCC; some estimates have even suggested that only two "medium scale" computers would be adequate for all the required functions. The straightforward substitution of one satellite processor for one or two computers is not valid in these cases, since each computer is being utilized for many functions besides track-correlation and coordinate conversion. Unless we can argue that the introduction of a satellite processor will actually eliminate one or more computers, or will permit us to perform the same job with a smaller computer, our claims for hardware cost savings are weakened. The example above in which we could save over $600,000 probably represents an extreme position.

The use of special-purpose hardware to reduce the cost of software development is a controversial subject at best. Many experienced software designers do not believe that special hardware offers any potential for savings, and bidders for military software contracts are probably not inclined to experiment with new approaches at the present time. The controversy seems to focus on two principal assertions by advocates of the "conventional" large-computer approach: First, they suspect that there are large hidden costs associated with the specification, design, programming (of the special devices), debugging, testing, integration, maintenance, and modification of special hardware. Second, they point to significant recent improvements in the art of software design--"top-down" and "structured" programming techniques, primarily--which offer the same organizational benefits of careful discipline and partitioning that have been employed in the hardware-design field for decades.

Resolution of the general controversy is well beyond the scope of this document. By suitably restricting our assumptions for the particular case at hand, however, we may gain some insight into the problem. Let us assume that the system designer has already studied the system requirements and has decided to implement the system in the form of a single central computer, without any satellite processors. Further, let us suppose that he has already chosen a computer which is large enough and fast enough to do the entire job. Finally, we imagine that he has performed studies on algorithms for coordinate conversion, track correlation, mapping, and masking, and that he knows exactly what he needs from each process by way of accuracy and timing. Having gone this far, we now offer the designer a choice: He may develop the software for his system as it stands, or he may add a satellite processor and then develop the software. Under these circumstances, will the addition of a satellite processor save money in the development of the software? Note that here the designer is not free to change his original computer selection.

It is helpful to consider the processing load placed on the computer by the requirements for mapping, masking, coordinate conversion, and track correlation. Without going into detail, we can guess that each radar detection entering the system (at an average rate of 1400 reports per second) will require five computer instruction-cycles to perform the mapping and masking functions; thus about 7000 instructions per second would be used for mapping and masking. Under the assumption that our computer uses software subroutines in the calculation of sines,

cosines, and square roots, a brief study shows that we might use 68 instruction-cycles to perform the coordinate conversions on one radar report; at an average report rate of 600 per second*, the coordinate-conversion process will consume about 41,000 instructions per second.

Estimates for the coarse-track correlation process are complicated by the large number of different approaches that might be used in the algorithm. The straightforward approach would test each of the 3600 reports against each of the 200 tracks, for a total of 720,000 tests. To test whether a given radar report is within 12 NM of a given track, it is necessary to execute about 12 instructions. A more efficient method would be to first subdivide the whole geographic coverage area into "strips" or "blocks", and then sort both the radar reports and the tracks into the various sub-areas; in this way we could avoid some unnecessary tests. For example, if we used five "strips", then we would need to make only 144,000 tests, at the expense of some additional bookkeeping. With five "strips", a total of about 1,800,000 instructions must be executed to perform coarse-correlations on all 3600 reports, but these tests must be completed within the last three seconds of each frame, for a peak rate of approximately 600,000 instructions per second. Other coarse-track algorithms might be devised to reduce the requirements on the computer, but it is clear that track-correlation represents a heavy processing load.

---

*After mapping and masking, the "raw" input reports are reduced to 600 reports per second for subsequent processing.

In Section III we discussed the procedures by which the satellite processor could be programmed, debugged as a unit, and tested with the central computer. If these procedures are as effective as we hope, then the cost associated with the development and testing of the satellite processor should be small compared to the total cost of the JSS. Let us assume that processor-development costs can be neglected.

In terms of the assumptions above, our question can now be restated: Will the addition of a zero-cost satellite processor (which will probably relieve the central computer of performing more than 600,000 instructions per second) save money in the development of the software? Potential cost-savings in software development will be reflected in the size of the computer program, and in the difficulty of writing and testing the program. These two aspects will be discussed below.

Based on experience with the Back-Up Interceptor Control (BUIC) system and other related multiple-radar tracking systems, we estimate that the radar-input portion of the JSS software might contain between 2000 and 3000 instructions (including mapping, masking, and coordinate conversion) and that the coarse-correlation software might contain 1000 instructions. Therefore the size of the overall software system will be reduced by perhaps 3000 to 4000 instructions using a satellite processor--a reduction of only a few percent to the total program. One way to estimate a cost savings is to assign an average cost for a single "finished" or "polished" instruction, and multiply this figure by 3000 or 4000 instructions. At (say) $30 per "polished" instruction,

the cost-reduction estimate would be $120,000 which, although not large in absolute terms, is nevertheless much more than the cost of the satellite processor. The difficulty with this form of estimation is that there are too many variables implicit in determining an average per-instruction cost, and the final estimates consequently lack credibility.

Although quantitative data are sparce, it is generally believed that the total cost of a computer program is a strong function of the degree to which the computer's capacity is stressed. This is especially true for programs which must operate in real time. Thus the closer a program must operate to the machine-dependent limits imposed by the computer's speed and memory, the higher the programming costs; and as these limits are approached, the cost increases rapidly. As examples, real-time software is thought to be five times more costly than non-real-time software, on the average[1], and programs which demand 90 percent of a computer's speed and memory capacity may cost three times more than routines which utilize only 50 percent of capacity[2].

With this in mind, we may view the satellite processor as a way to retreat from the limitations imposed by our choice of a computer--to move away from the machine's performance thresholds, and out of the high-cost region. It is easy to imagine that the addition of an advanced satellite, which can relieve the JSS computer of up to 600,000 instructions per second, might move the software problem from the 90 percent to the 50 percent region of machine capacity, and we could credit the satellite

with reducing software costs by a factor of three, or more than one million dollars.

This kind of cost-saving estimate serves merely to suggest the impressive potential of the satellite processor in certain cases. It also shows that such savings are influenced by the choice of the central computer: If the computer is selected to be just barely large enough and fast enough for the JSS problem, then the addition of a satellite could save perhaps two-thirds of the programming costs; but if the computer is initially operating well below its capacity (a more expensive computer), then the satellite might not strongly affect the cost of writing the software.

In summary, we have not succeeded in making quantitative estimates of the satellite's impact on the cost of the JSS. Subject to a variety of assumptions and constraints, we have argued that a satellite processor might save as much as $600,000 in direct hardware savings. With an even more restrictive set of assumptions, we estimated software savings of more than one million dollars. These estimates are probably mutually exclusive, i.e., the satellite processor would be unlikely to demonstrate large savings in both hardware and software simultaneously. The real point of the exercise is to establish the potential for significant cost savings far beyond the development cost of the processor.

SECTION VI

CONCLUSIONS

We believe that special-purpose processors, used in "satellite" configurations with a general-purpose computer, can represent a powerful way to simplify the overall system architecture, lower the equipment costs, make the computer programming easier and less expensive, and greatly expedite testing and debugging.

Satellite processors are especially well suited for mathematically simple functions that must be repeated rapidly and often. The conversion of radar reports from polar coordinates to rectangular coordinates is one such function; the correlation of radar reports with target tracks is another.

The Joint Surveillance System (JSS) was selected here for study because it offers the typical mixture of equipment and computational problems found in relatively large data-processing centers. A "baseline" satellite processor, designed to perform coordinate-conversion and track-correlation in real time, was worked out in detail to provide a specific example in terms of hardware and software complexity, size and cost, and interfaces with the central computer.

The potential cost benefits to JSS will, of course, depend heavily on the particular system configuration and on a large number of other assumptions, e.g., the manner in which the system tasks are subdivided among the various computational elements. For one set of assumptions,

the use of an "advanced" satellite processor (whose total cost, including development, is considerably less than $50,000) could result in a net savings of nearly one million dollars.

The processor described in this document has been specialized to the JSS requirements, but the general techniques used in its design can be applied to other track-while-scan radar surveillance systems. It is likely that similar devices could be equally attractive for use with FAA radars, the AWACS airborne surveillance platform, the Conus Over-the-Horizon radar, and a variety of tactical radars.

APPENDIX A

ESTIMATES OF MICROPROCESSOR PROGRAM SIZE AND TIMING

The major tasks to be performed by the microprocessor in the "baseline" satellite are the conversion of the raw reports from polar coordinates to cartesian coordinates, and the calculation of position differences between the radar reports and the target tracks (in support of the coarse track-correlation operation). The programming considerations for these two processes are described in more detail below:

Coordinate Conversion

Given a radar report consisting of a slant-range measurement R, an azimuth measurement θ, and a height measurement H, it is desired to perform the following calculations to produce the X- and Y-values of the report:

$$\text{Let} \quad F = \sqrt{R^2 - (H-K_1)^2}$$

$$\alpha = \theta - K_2$$

$$\beta = 2\alpha - K_3$$

then

$$X = K_4 + K_5 F \sin\alpha + K_6 F^2 \sin\beta$$

$$Y = K_7 + K_8 F \cos\alpha + K_9 F^2 \cos\beta$$

where K_1 , K_2 , ... K_9 are site constants.

Since the microprocessor selected for our study does not have a hard-wired capability for multiplication, we will use a software multiply-routine, and choose algorithms which avoid division.

The coordinate-conversion program will use four subroutines: multiplication, sine, cosine, and square root. Particular algorithms optimized for the microprocessor have not been worked out in detail, but we have selected representative methods for performing these functions for the purpose of estimation. The subroutine for multiplication will require about 20 program instructions, and will consume about 20 machine cycles during execution. The sine subroutine is based on a four-term power-series expansion; it will require about 20 program instructions and four constants, and will take about 170 machine cycles to execute. The square root subroutine employs a combination of table lookup and power-series expansion; its program occupies about 30 instruction locations and uses 32 constants, and it requires about 40 machine cycles to execute. Finally, the cosine subroutine evaluates the expression

$$\cos \phi = \pm \sqrt{1 - (\sin \phi)^2}$$

It requires about 10 program instructions and takes about 70 machine cycles to execute. In summary, the subroutine characteristics are shown below:

Subroutine	Instructions	Constants	Execution Time
Multiplication	20	--	20 cycles
Sine	20	4	170
Cosine	10	--	70
Square Root	30	32	40
	80 total	36 total	

Table A-1 shows a version of the program that might be suitable for the coordinate-conversion manipulations. The program contains 44 instructions, and requires 768 machine cycles to execute; it also requires temporary storage for variables T, F, F^2, $\sin\beta$, $\cos\beta$, and $\sin\alpha$, some or all of which can be accommodated within the available 16 general-purpose registers in the microprocessor itself.

For purposes of estimation, we shall assume that the total coordinate-conversion program requires 124 program instructions (80 for subroutines and 44 for the program itself) and storage for 36 constants. If we allow some margin--e.g., to execute various shifting operations to avoid the loss of precision with fixed-point arithmetic--we may estimate that the overall program would require 800 machine cycles to complete the coordinate conversion of one radar report.

Coarse Track-Correlation

As discussed in Section III, the process of coarse track-correlation can be implemented through a combination of general microprocessor operations and special hard-wired logic operations.

The microprocessor is expected to calculate two arithmetic differences and deliver them to the hard-wired device. The two values desired from the microprocessor are ΔX and ΔY, the difference between the X-value of the radar report and the X-value of the target position, and the corresponding difference for the Y-coordinates of the report and the target. Since the hard-wired logic can perform its function in one clock cycle, the timing constraints are set by the microprocessor.

Table A-2 shows a version of the sub-program that might be suitable for the calculation of ΔX and ΔY, and their delivery to the hard-wired logic.

We have taken advantage of the microprocessor's internal registers and their capability for automatic incrementing within an instruction. The principal time-consuming computational loop involves six instructions and requires six machine cycles to complete. Since the satellite must test each radar report against as many as 200 target tracks, we can estimate a maximum of 6 x 200 or 1200 machine cycles for the coarse correlation of one radar report. This ignores the time required to set up the various microprocessor registers and perform other housekeeping functions, but these operations represent a very small fraction of the total time used in coarse track-correlation.

INST. NO.	INSTRUCTION	NO. OF STEPS
1	FETCH H	1
2	SUBTRACT K_1	1
3	STORE IN T	1
4	MULTIPLY BY T	20
5	STORE IN T	1
6	FETCH R	1
7	MULTIPLY BY R	20
8	SUBTRACT T	1
9	CALL SQ ROOT	40
10	STORE IN F	1
11	MULTIPLY BY F	20
12	STORE IN F^2	1
13	FETCH θ	1
14	SUBTRACT K_2	1
15	STORE IN T	1
16	MULTIPLY BY 2	1
17	SUBTRACT K_3	1
18	CALL SINE	170
19	STORE IN SINβ	1
20	CALL COSINE	70
21	STORE IN COSβ	1
22	FETCH T	1

INST. NO.	INSTRUCTION	NO. OF STEPS
23	CALL SINE	170
24	STORE IN SINα	1
25	CALL COSINE	70
26	MULTIPLY BY F	20
27	MULTIPLY BY K_8	20
28	STORE IN T	1
29	FETCH COSβ	1
30	MULTIPLY BY F^2	20
31	MULTIPLY BY K_9	20
32	ADD T	1
33	ADD K_7	1
34	STORE IN Y	1
35	FETCH SINβ	1
36	MULTIPLY BY F^2	20
37	MULTIPLY BY K_6	20
38	STORE IN T	1
39	FETCH SINα	1
40	MULTIPLY BY F	20
41	MULTIPLY BY K_5	20
42	ADD T	1
43	ADD K_4	1
44	STORE IN X	1

TABLE A-1. COORDINATE TRANSFORMATION PROGRAM.

1. Load $X_R \rightarrow R_1$
2. Load $Y_R \rightarrow R_2$

} radar report values

3. Load starting address of track table $\rightarrow R_3$
4. Load $N \rightarrow R_4$ (N = no. of tracks)
5. Subtract X-value of track (address in R_3) from R_1, leaving answer in accumulator. Increment R_3.
6. Output accumulator (ΔX) to hardwired logic.
7. Subtract Y-value of track (address in R_3) from R_2, leaving answer in accumulator. Increment R_3.
8. Output accumulator (ΔY) to hard-wired logic.
9. Decrement R_4, skip next instruction if result is zero.
10. Jump to instruction #5.
11. Continue to remainder of program.

R_1, R_2, R_3, and R_4 are internal microprocessor registers. Instructions #1-#4 set up a computational loop consisting of the six instructions #5-#10. When the program exits to instruction #11, it has completed the coarse correlation of one radar report against N target tracks.

TABLE A-2. SUB-PROGRAM FOR COARSE TRACK-CORRELATION.

APPENDIX B

PROGRAMMING CONSIDERATIONS

This Appendix will describe how the microprocessor program for the "baseline" satellite processor proceeds from one function to another. In general, the program contains several decision points which compare the counts in various registers and also the status of the I/O control lines. Direct Memory Access (DMA) is performed upon demand by the central computer and is carried out using the central computer clock. Upon completion of DMA, the satellite processor clock is restored, and the program resumes from the point of interruption.

Reference Address Registers

The program developed for the satellite processor progresses according to the amount of data received and the amount of radar data processed. This is accomplished by regular referencing to various address registers. These address registers are listed in Table B-1.

To accommodate the overlap in frames (when data from the previous frame are being returned to the computer while new radar data are being received), two separate radar data address counters are used, one for even-numbered frames, and one for odd-numbered frames. "Odd" and "Even" are arbitrarily determined by a modulo-two counter that counts frame-start synchronization signals. For example, counter "a" will count all radar data entered during the first (odd) frame. Then counter "b" will count all radar data in the next (even) frame.

"a"	last radar data address entered from start of odd frame.
"b"	last radar data address entered from start of even frame.
"c"	last radar data address which has been coordinate converted.
"d"	last radar data address which has been correlated.
"e"	site data address counter.
"f"	current track data address.
"g"	last track data address.
"h"	last address of data transferred from satellite to computer.

Table B-1. Register Identification.

After all data have been returned to the computer from the first frame--which occurs within 0.5 seconds of the end of the frame--counter "a" is cleared and ready to count radar data in the next odd frame, which will begin in about 5.5 seconds.

Reference address registers (counters) "a" and "b", as appropriate, are continually incremented throughout their respective frames as new radar data are received. As these radar data are coordinate-converted, register "c" is incremented to indicate how many radar data have been coordinate-converted during each frame. Similarly, register "d" is incremented once every time one coordinate-converted radar data point has been correlated with the radar track data.

Register "e" holds the address corresponding to the site number, so that the appropriate site data can be addressed during the coordinate conversion. Register "f" contains the address of the current radar track data during the correlation subroutine. Register "g" contains the last radar track address received from the computer. This is used to terminate each correlation subroutine by comparison with register "f".

Register "h" contains the address of the last data transferred to the computer from the satellite processor. When compared with register "a" or "b", as appropriate, this indicates when all data have been returned to the computer, and that the register (counter)

can be cleared. This condition affects the re-set command if the new frame synchronization pulse has been received.

The frame-start and radar-data counter select procedure is shown in Figure B-1. Upon the occurrence of the frame signal, a circuit will alternately select registers "a" and "b" for counting radar input data. The appropriate counter will be cleared after "reset", as depicted in Figure B-2.

Figure B-2 is the program description for performing coordinate conversions and track correlations. Upon the generation of a re-set command, counters c, d, e, f, g, and h are initialized. If counter "a" has been enabled by the counter "a/b" selection circuit, then counter "b" is cleared; if counter "b" has been selected, then counter "a" is cleared. Next, if the content of the uncleared counter, above, exceeds the number of coordinate conversions executed (indicated by counter "c"), then a coordinate conversion is executed on the data corresponding to the count (address) in "c", after which "c" is incremented by one count and the program returns to point A.

If the coordinate-conversion sequence has caught up with the input data, and if all track data have been received, and if coordinate-converted data exist which have not been correlated (c>d), then the correlation sequence will be entered for the address in counter "d". When a correlation sequence has been completed, counter "d" is incremented by one and the track data address counter is initialized,

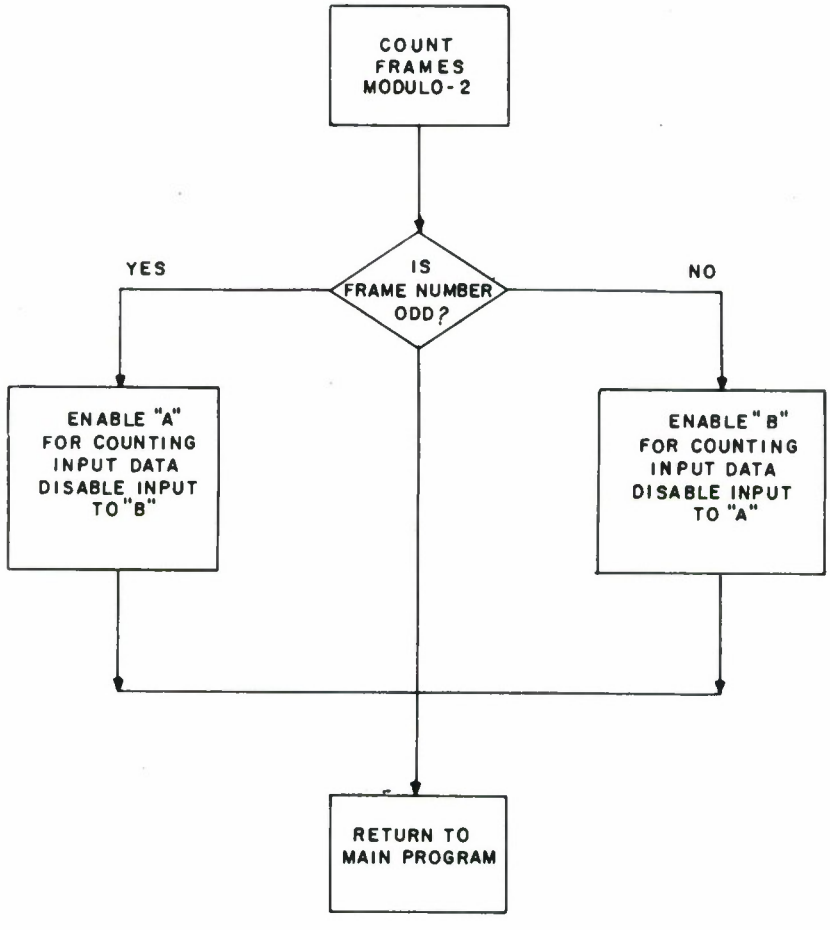


Figure B-1 COUNTER "A"/"B" SELECTION

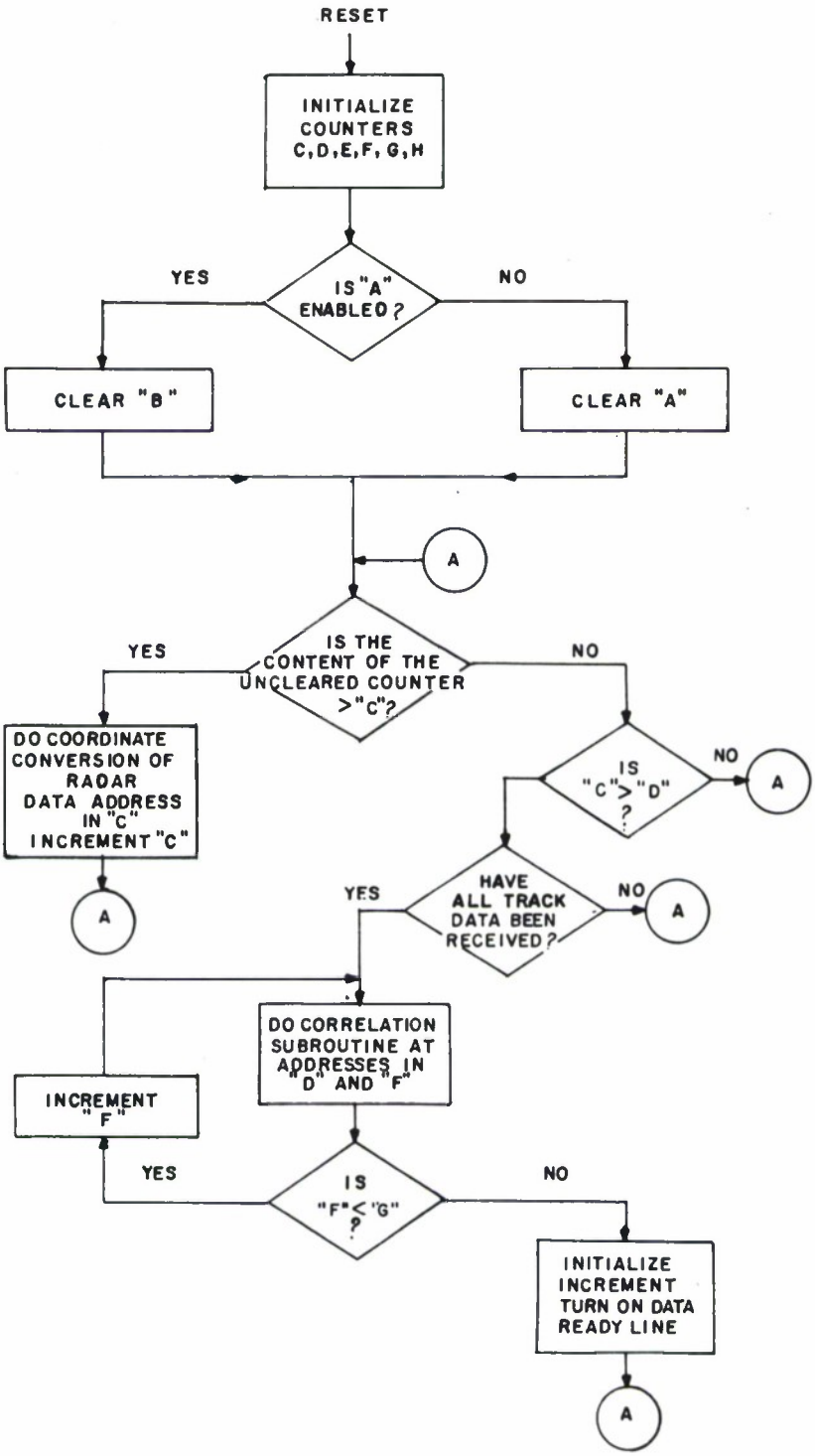


Figure B-2 FLOW DIAGRAM FOR COORDINATE CONVERSION AND TRACK CORRELATION

to be ready for the next correlation; the program is directed to point A so that coordinate conversions can be carried out on any additional radar data that may have been received in the interim. If no more data were received, then the program will advance to the correlation sequence for the next address in "d", etc. Whenever a correlation process has been completed, the "Data Ready" flag to the computer will be raised. When the satellite processor has transmitted all data prepared for the computer at any time, the "Data Ready" flag will be dropped. Thus the computer can transfer data from the satellite processor at will, whenever the "Data Ready" flag is raised. This flag will typically be raised and lowered many times during the last three seconds of each frame if the computer requests data as they are generated. By delaying its request to just after the beginning of a frame, all data could be transferred during a fairly short time period. The satellite will automatically accommodate either mode of data transfer.

DMA Operation

All data transfer can be effected by DMA. The previously mentioned control lines between the satellite processor and the computer will determine whether the satellite should input or output data, and, if inputting data, whether they are radar data, track data, or site data. Also required is a line from the central computer indicating that track data transmission is finished, and lines from the satellite processor acknowledging the DMA request

and indicating that it has data ready to transfer. Other lines from the computer will indicate beginning-of-frame, and will provide the data strobe to the satellite processor for inputting or outputting data.

Figure B-3 shows the general operation of the DMA. DMA can occur on any machine cycle of the satellite processor, which is 330 nanoseconds. Therefore, the satellite processor will respond to a DMA request within, at most, several machine cycles. This is achieved by simply re-synchronizing the DMA request with the satellite processor clock, and AND-ing this with the appropriate microprocessor controls. The proper control lines must, of course, be set by the time the DMA request is issued. Data will be clocked in or out under control of the computer. A one-megahertz clock is assumed, although this is not very critical. As data are entered or outputted, appropriate counters will be incremented to keep track of the data transfers. When the DMA request line returns to normal, the satellite processor will continue from the point at which it was stopped.

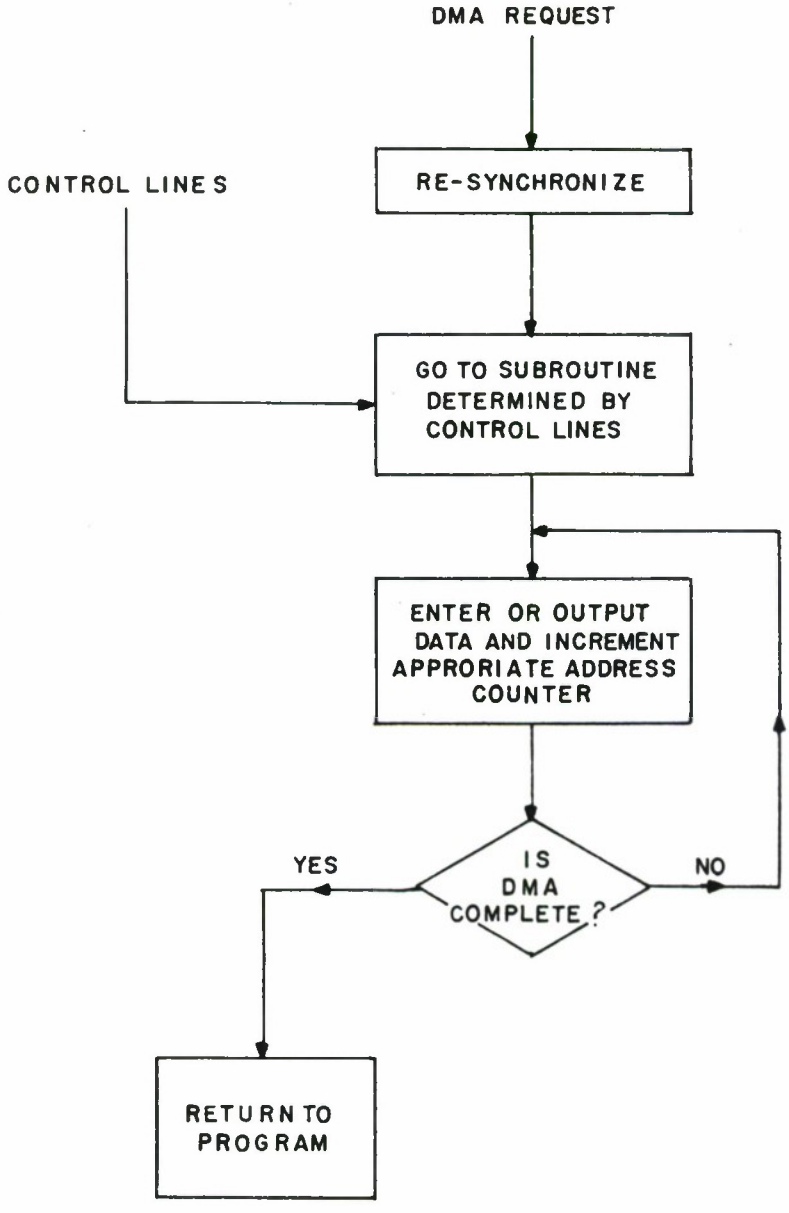


Figure B-3 FLOW DIAGRAM FOR DMA PROCEDURE

REFERENCES

1. Clapp, Judith A., "A Review of Software Cost Estimation Methods," ESD-TR-271, Electronic Systems Division, AFSC, Hanscom AF Base, Mass., August 1976.

2. Boehm, B. W., "The High Cost of Software," Practical Strategies for Developing Large Software Systems, E. Horowitz, Ed., Addison-Wesley Publishing Company, Reading, Mass., 1975.